Determining Fuzzy Link Quality Membership Functions in Wireless Sensor Networks

By

Syed Ali Hussain Kazmi

A Thesis Submitted in Partial Fulfillment

Of the Requirements for the Degree of

Master of Applied Science

In

Electrical and Computer Engineering

Faculty of Engineering and Applied Science

University of Ontario Institute of Technology (UOIT)

Oshawa, Ontario, Canada

April, 2014

ABSTRACT

Wireless Sensor Network routing protocols rely on the estimation of the quality of the links between nodes to determine a suitable path from the data source nodes to a data-collecting node. Several link estimators have been proposed, but most of these use only one link property. Fuzzy logic based link quality estimators have been recently proposed which consider a number of link quality metrics. The fuzzification of crisp values to fuzzy values is done through membership functions. The shape of the fuzzy link quality estimator membership functions is primarily performed leveraging qualitative knowledge and an improper assignment of fuzzy membership functions can lead to poor route selection and hence to unacceptable packet losses.

This thesis evaluated the Channel Quality membership function of, an existing fuzzy link quality estimator and it was seen that this membership function didn't perform as well as expected. This thesis presents an experimental approach to determine a suitable Channel Quality fuzzy membership function based on varying the shape of the fuzzy set for a multipath wireless sensor network scenario and choosing an optimum shape that maximizes the Packet Delivery Ratio of the network. The computed fuzzy set membership functions were evaluated against an existing fuzzy link quality estimator under more complex scenarios and it is shown the performance of the experimental refined membership function was better in terms of packet reception ratio and end to end delay.

The fuzzy link quality estimator was applied in WiseRoute (a simple converge cast based routing protocol) and shown that this SNR based fuzzy link estimator performed better than the original implemented RSSI based link quality used in WiseRoute.

ACKNOWLEDGEMENT

At this point I would like to express my deep gratitude to my supervisors Dr. Ramiro Liscano and Dr. Jing Ren for their support and guidance throughout my thesis. They were always very helpful, motivating and understanding and I learned a lot from them.

I owe a special thanks to my friends especially Nisha, Visal, Kelsey, Nadra, John, Brent, Krupa, and Faizan for their support and encouragement during difficult moments of my thesis and for the environment in the lab.

I would also like to thank Dr. Kamran Sartipi and Dr. Shahram Heydari for providing this environment with Dr. Ramiro Liscano in our lab and also helping and guiding us when needed be it research or coursework. They both made me feel like I am one of their own students.

Finally I would like to convey my sincere and deep gratitude to my parents, my sisters, brothers in law, my nephew and nieces. Without their patience, encouragement and support, I could not have completed my thesis. Their faith in my abilities was really encouraging.

Table of Contents

List of Figures

Chapter 1 Introduction

1.1 Wireless Sensor Networks

A Wireless sensor network is a collection of sensor nodes spread across an area of observation to observe the surroundings and collect data. This data can be temperature, motion, pressure, etc. Sensor nodes are low cost, low powered, tiny devices with less processing, limited energy, and limited memory and transmission capabilities [4]. The sensor networks are usually deployed in harsh environments where it is too dangerous or impossible for human beings to work. These sensor nodes observe and report any changes in external environment. These sensor nodes are capable enough to run routing protocols and communicate between each other. The data collected by the sensor nodes over time is then transmitted to a base station or actuator for further processing and/or accordingly take actions. The wireless sensor network is supposed to operate unattended for a long period of time due to the extreme conditions or applications it is often used in.

Usually a wireless sensor network is composed of four components [2]. These components are *(i)* **Source Nodes**, *(ii)* **Sink Nodes**, *(iii)* **Wireless Links** and a *(iv)* **Central Processing Node**. A Source node is a node which is responsible for observing the environment and collecting the data. A Sink node on the other hand is receiving the data. The wireless link is the channel through which two nodes communicate between each other. The Central Processing node is usually outside the usual sensor field and has higher computation and processing capabilities and also has the decision making authority. The Central Processing Node is usually the node where user can also access the

data and is connected to a sink node. A source node generates a data packet by detecting an unusual event in the environment and using a communication channel that data packet is delivered to a sink node. Any number of source nodes can exist, which are collecting the data and then transmitting those data packets to one or several sinks.

Following are the terms related to wireless sensor networks used in this thesis.

a. Sensor Node: A sensor node is a device with a number of sensors installed into it. These sensors observe the environment and any changes are reported. An example can be a sensor node which can sense heat and humidity.
b. Source Node: A source node is a sensor node which is responsible to observe the environment and collect data.
c. Sink Node: A sink node is a sensor node which is responsible to collect all data it receives from one or many source nodes.
d. Route: The path a packet uses to reach the sink node from the source node is a route in a wireless sensor network.
e. Data Packet: A data packet is a packet which has the information collected by source node and is supposed to use the route from source node to sink node.
f. Beacon: A beacon is a packet which is initiated by the sink node. It is used to create a routing tree from a source node to sink node.
g. Routing Tree: A routing tree is a link from source to sink node, in which all nodes use a route to connect to sink node using multi-hop communication.

h. Parent Node: Like the name, a parent node in a routing tree is next closest node which is also closer to sink node. It will take 1 less hop from this node to reach sink node.

i. Link Estimator: A link estimator is a way of estimating the link quality and tells the routing protocol which route has best quality to sink node. Several routing estimators are available for routing protocols to use and estimate the link towards sink and transmit the data using that link.

The delivery of the collected data from source nodes to sink node in a Wireless Sensor Network (WSN) is done wirelessly. Usually the nodes, which are not acting as source nodes or sinks are used as routers; although the role of routers can also be played by different source nodes as well. So a node can have dual role of source node and due to its location it can also be a forwarding node.

Sensor nodes have only local knowledge of the network and that's why multi-hop communication is needed. Not all sensor nodes are directly connected to a sink node due to the radio transceiver and energy shortcomings [3]. The data communication in a Wireless sensor network is multi-hop communication. A sensor node collects the data and using different routing strategies the data is then delivered to sink by multi-hop communication.

The multi-hop communication in wireless sensor networks can be unreliable [1]. If the route towards sink is not selected based on a proper strategy, the data can be lost in the network or can create congestion in the network. The data must be delivered through best route in a way that minimum amount of energy is consumed. Keeping the energy

shortcomings in mind and due to unreliable communication, the optimum route for transmission of data must be selected. The appropriate route for transmission can be the route which has least end-to-end delay, least hop count, maximum packet reception ratio, less energy consumption, fault tolerance, and avoiding the creation of any energy holes in the network.

1.2 Problem Statement

Link estimation is one of the most important attributes of routing protocols for wireless sensor networks and bad link estimation may lead to less stable networks with packet loss or delay or both. Recently a fuzzy logic based link estimator, labeled as F-LQE, was proposed [15] and it demonstrated to perform better than other popular link estimators when applied to a collector tree routing protocol. The membership functions of this algorithm were both qualitatively and quantitatively determined but not optimized. The objective of this thesis was to determine an approach to calculate a refined fuzzy membership function for WSN.

1.3 Thesis Contribution

The contribution of this thesis is an algorithm for the optimum selection of a fuzzy membership function for a WSN. A three step simulation strategy was used to determine the membership function. The fuzzy link estimator was applied in WiseRoute and after a number of simulations a good membership function was assigned and compared to the membership functions used in F-LQE.

The results showed that the approach in this thesis was better than other strategies including WiseRoute based on fuzzy link estimator with four link properties and Original WiseRoute.

1.4 Thesis Structure

This thesis is organized into six chapters. First chapter gives introduction to problem and thesis contributions. Second chapter gives overview of number of link estimators and routing protocols used and how link estimators are used in number of routing protocols. Fuzzy logic based link estimators are also introduced in this chapter.

In chapter 3 the proposed three step strategy to determine the fuzzy membership function was discussed and how membership functions effect the link selection in wireless sensor networks.

Chapter 4 shows the implementation of fuzzy logic based link estimator in WiseRoute. Chapter 5 presents the results of number of strategies used and compared to fuzzy logic based link estimator in WiseRoute.

Finally chapter 6 states the conclusion and future work.

Chapter 2 Related Works

This chapter discusses some of the commonly used routing protocols and link quality estimators in WSNs. The focus will be on converge cast routing protocols. Some of the commonly used Link Quality Estimators, link quality estimators using multiple properties and those link quality estimators which use fuzzy logic for combination of number of link quality estimators will be discussed.

2.1 Link Quality Estimators in Wireless Sensor Networks

This section discusses a number of link quality estimators some of which are later optimized for membership function in fuzzy link estimation. In wireless sensor networks routes can be very unreliable due to harsh environment, the application specific problems and interference of signals. The radio transceivers in wireless sensor networks are not very efficient. These transceivers transmit a weak signal which is prone to distortion from other nodes. When data is transmitted, the data packets due to weak signals are modified and make the communication very unreliable. Due to all these problems data packets should be sent through the route which is more reliable in terms of Packet Reception Ratio and less hops to preserve energy.

In Wireless Sensor Networks the routes towards sinks are evaluated using link quality metrics also known as Link Estimators. Link Estimators help choose the route towards a sink using different properties of route. A link estimator chooses one of the neighboring nodes as parent node and transmits data towards sink using that node. The node, a link estimator chooses, is supposed to be the best in terms of quality. For example a link estimator can choose a route towards sink which has maximum packet reception ratio or a

link which consumes least amount of energy or which takes least hops for a packet to reach the sink node. If the links are estimated accurately, the route for packet delivery will be more reliable in terms of packet delivery ratio and stability [12].

A link quality estimator uses a routing metric which is used to evaluate a route towards a sink node from source node. The route is evaluated by evaluating the link between each neighbor. The link quality is measured by using one or more than one link quality metrics. The Link Quality Estimators are classified in two categories [12], Hardware based Link Quality Estimators and Software based Link Quality Estimators. Figure 2.1 shows the classification of different types of link quality estimators.

2.1.1 Hardware based Link Quality Estimators.

Hardware based link quality estimators are directly read from the Radio transceiver. Hardware based link quality estimators do not need any computation but they are usually termed as not giving as good estimate as any software based metrics.

2.1.1.1 Signal to Noise Ratio (SNR)

SNR is the ratio of the power of the signal to the noise level. It measures how much signal is corrupted by noise. So SNR measures the quality of link by measuring the noise in the wireless link.

2.1.1.2 Received Signal Strength Indicator. (RSSI)

RSSI is a link quality estimator which is received directly from the radio transceiver. RSSI is calculated by sampling the signal strength of the first 8 symbols after the Start of

Frame Delimiter (SFD) in a packet. Being hardware based metric RSSI gives a quick estimate of a link if it's in grey area also known as transitional area.

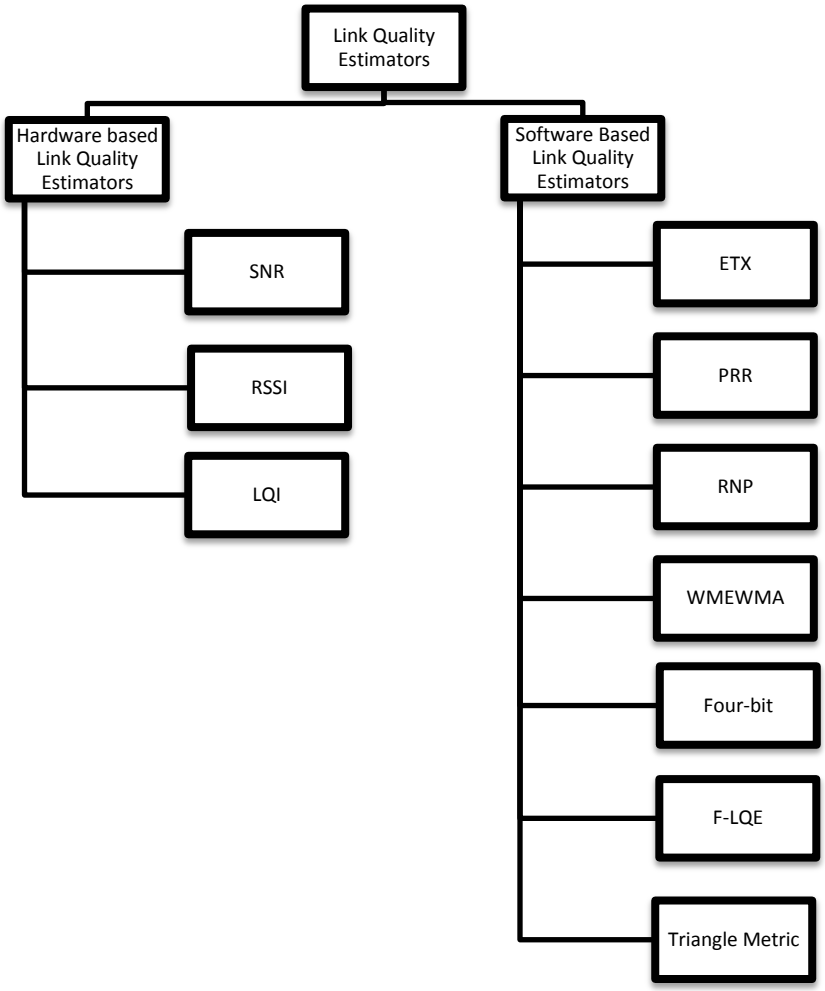

Figure 2.1: Types of Link Quality Estimators

2.1.1.3 Link Quality Indicator (LQI)

A hardware based link estimator, Link Quality Indicator is provided by the CC2420 radio. LQI is computed by received signal strength and number of errors received. Like

RSSI, LQI is an average correlation value of samples computed over first 8 symbols following the start of frame delimiter

2.1.2 Software based Link Quality Estimators

Software based link quality estimators are computed by the number of received and sent packets. Different strategies are adopted to calculate software based link quality estimators. Some software based link quality estimators are calculated at sender side, while others are calculated at the receiver side.

2.1.2.1 PRR [12]

Packet reception Ratio (PRR), a receiver side estimator, is calculated by Number of successfully received packets over Number of sent packets. The number of received packets is sum of all packets transmitted by source node, Packets which are received and those which are lost too.

$$PRR(\omega) = \frac{Number\ of\ Received\ Packets}{Number\ of\ Sent\ Packets}$$

Where (ω) is the estimation window.

2.1.2.2 ETX [23]

ETX is an acronym for Expected Transmission Count. It is a receiver side estimator and it approximates the number of retransmissions are required to successfully deliver a packet [13]. It takes link asymmetry into account and is computed by combining PPR for uplink and PRR for downlink. Asymmetry is the bidirectional link quality, i.e. the link quality from node A to node B and link quality from node B to node A both taken in

consideration. It checks how many packets are sent before receiving an acknowledgement for that packet.

$$ETX(\omega) = \frac{1}{PRR_{forward} \times PRR_{backward}}$$

Where ω is the estimation window; $PRR_{forward}$ is the packet reception ratio from sender to receiver and $PRR_{backward}$ is the PRR from receiver to sender.

2.1.2.3 RNP

Acronym for Required Number of Packets, RNP is calculated by the required number of transmissions and retransmissions for successful delivery of a packet. It is a sender side estimator and uses acknowledgement messages to determine if a packet is successfully transmitted.

$$RNP(\omega) = \frac{Number\ of\ Transmitted\ and\ retransmitted\ Packets}{Number\ of\ Successfully\ received\ packets} - 1$$

2.1.2.4 WMEWMA [24]

Acronym for Windows Mean Exponentially Weighted Moving Average, WMEWMA uses EWMA [24] filter to combine new and old PRRs to remove fluctuations and smoothen the PRR. It uses a smoothing factor α which fluctuates between 0 and 1.

$$WMEWMA(\alpha, \omega) = \alpha \times WMEWMA + (1 - \alpha) \times PRR$$

Where α is the smoothing factor and ω is the estimation window.

2.1.2.5 Four-bit [14]

Four-bit provides link information in the form of four bits. White bit is from the Physical layer, Ack bit is from the Link layer and Pin bit and Compare bit are from the Network layer. White bit shows that whether the link is of high quality or not which allows the estimator to avoid bad link for data transmission. The Ack bit keeps track of whether the acknowledgement for a sent packet is received or not. Compare and Pin bits are used to compare a link with the ones already in link table and whether a better link is available. Pin bit is used to pin the current link in link table and not to remove it.

2.1.2.6 F-LQE [15]

A recently proposed F-LQE Fuzzy Logic based Link Quality Estimator combines four properties using fuzzy logic [15].

This estimator considers four link quality properties namely packet delivery, asymmetry level, channel quality, stability factor. Three of four link properties are calculated using packet reception ratio and the fourth link property, the Channel Quality, is directly received from the radio transceiver using SNR. These link properties are combined using a fuzzy rule

> "**IF** the link has high *packet delivery* AND low *asymmetry* AND high *stability* AND high *channel quality* **THEN** it has high quality."
>
> Then the values are smoothened using EWMA filter [24].

If the rule is satisfied the link is selected. Detail of F-LQE can be found in coming chapters.

2.1.2.7 Triangle Metric [21]

Triangle metric geometrically uses three link properties, one software property and two hardware properties, to estimate the link. Triangle metric requires PRR, SNR and LQI and geometrical combination to compute an estimate of the link [21]. The authors claim that triangle metric can estimate the link with as few as 10 packets for a reliable link and can be used in both static and mobile environments.

2.1.2.8 FLI [22]

Fuzzy Link Logic Based Link Quality Indicator is a recently proposed link estimator. It uses Packet Reception Ratio (PRR), Coefficient of Variance (CV), and Distribution of correlation (D_α) to compute fuzzy link quality estimator. The two membership rules are:

(1) IF "PRR is HQ" and "CV is HQ" and D_α is HQ" THEN "FLI is HQ".

(2) IF "PRR is LQ" and "CV is LQ" and D_α is LQ" THEN "FLI is LQ" [22].

According to [22] the FLI based Collection tree Protocol (CTP) [11] performed better than 4-bit based CTP.

2.2 Types of Routing Protocols in Wireless Sensor Networks

The main focus of this work is on the optimization of fuzzy membership functions for link estimators in Wireless sensor networks. This work will discuss different types of routing protocols. A packet is created by a source node by sensing the environment and then it is transmitted to the base station. Packet transmission from one point to other is done using routing protocol. The packet should take the best possible route to reach the sink node. The route selection in a wireless sensor network is done using link estimator.

A routing protocol using number of link estimators finds the best route towards sink. The route which is best according to link estimator is then used for transmission of data from source to sink.

In the beginning of network, a routing protocol sends a route construction packet which is transmitted in the whole network to know the neighbors in the network. These nodes calculate the link quality and save the best possible node available in terms of quality and save that node for forwarding the packet towards sink. Several routing protocols are proposed which collect the data and forward the data towards sink node. Some nodes with higher transmission power and memory are selected as sink nodes. These nodes are used to collect data from different nodes in the network. Source nodes are used to collect data from the environment and transmit data towards sink node and number of forwarding nodes are used to forward the data from source to sink node and act as routers.

In [5] according to Gergely Ács, Levente Buttyán these routing techniques are divided into five categories: Content-based routing protocols, Probabilistic routing protocols, Location-based routing protocols, Hierarchical-based routing protocols, and Broadcast-based routing protocols.

2.2.1 Content based routing protocols

Also known as Data Centric Routing Algorithm the Content based routing protocols; choose the next hop based on the query sent by the sink node. In content based routing protocols, all nodes are requested for data and whichever node has the data should send the data. SPIN [8] and Directed Diffusion are examples of content based routing. CTP

(Collection Tree Protocol) [11] one of the most widely used routing protocol is also an example of data centric routing protocol.

Directed Diffusion as proposed in [6] is a content based protocol. In directed diffusion sink node floods the network with a request for specific data. All nodes once receive the request set the gradient to the node they received the request from. This node is used to send data towards sink node later. A link quality metric like delay is used to set the gradient and send the data. A node might receive request for data from multiple neighbors and set multiple neighbors with multiple gradients as its neighbors. So the path with least delay is selected.

2.2.2 Probabilistic Routing Protocols

Probabilistic routing protocols aim in load balancing for the selection of next hop towards destination in route and route is selected in a random manner. These protocols look for certain quality of service parameters like avoiding energy holes. Energy Aware Routing [7] is an example of probabilistic routing protocol.

In energy aware routing protocol, destination node initiates a routing topology. Each node randomly selects next-hop based on the quality. The lower the quality of the next hop, the lower the chance of that node to be selected as the next hop. The routing metric of Energy aware routing protocol is the energy in that node and the energy it requires to transmit the packet. All the information about the quality of the node is provided by the MAC protocol.

Energy aware routing protocol consumes more energy in the communication in the setup phase of the protocol.

2.2.3 Location-based Routing Protocols

Location based routing protocol selects the next hop towards sink based on the location of the nodes. The locations of nodes and sinks should be known to transmit the data towards sink. GEAR is example of Location based routing protocol.

2.2.4 Hierarchical- based routing protocols

Also known as cluster based routing protocols; Hierarchical-based routing protocols are distributed in clusters and have cluster heads. Number of nodes has one cluster head which aggregates the data and then transmit the data. This aggregation of data in one node and then forwarding conserves energy. Cluster heads are more resourced nodes and have higher communication capabilities and energy resources than other nodes. LEACH [10] and TEEN [9] are examples of Hierarchical based routing protocols.

2.2.5 Broadcast based routing protocols

Individual nodes decide to forward a packet or not. If a node decides that a packet should be forwarded, it simply rebroadcasts a packet. Otherwise a packet is discarded if a node decides otherwise. MCFA (Minimal Cost Forwarding Algorithm) is an example of broadcast based routing algorithm.

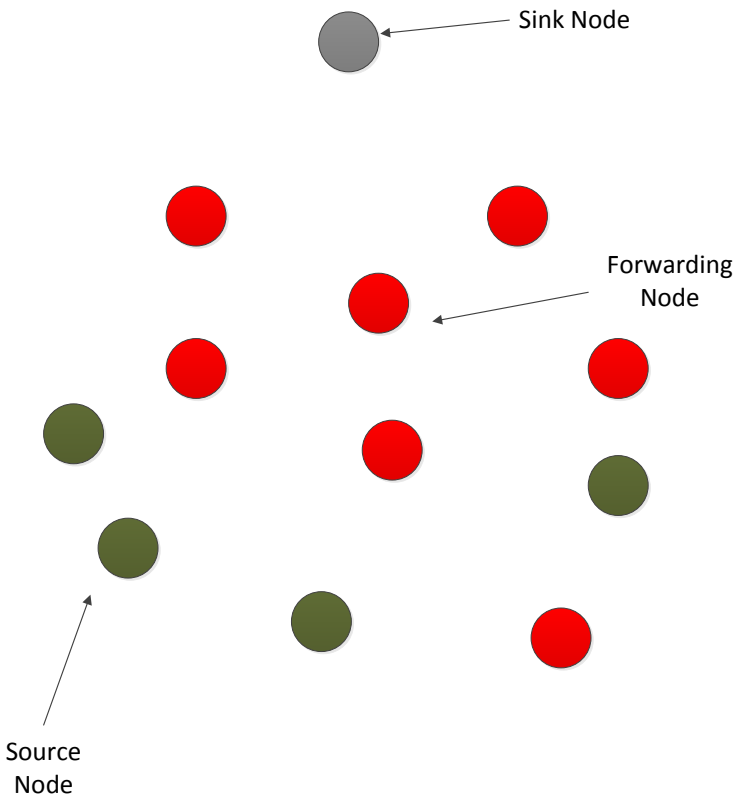

Figure 2.2: Different Nodes in Wireless Sensor Networks

2.3 Collector Routing Protocols in Wireless Sensor Networks

This section discusses the collector style routing protocols because the optimized link quality membership functions are used in converge cast based routing protocols. The converge cast based routing protocols use link estimators to estimate the links. Based on link estimation, route towards sink is selected and that route is then used for data transmission. Figure 2.2 shows different types of nodes in collector style routing protocols.

2.3.1 Collection Tree Protocol

Gnawali et al. proposed Collection Tree Protocol in [11]. It is one of the widely used tree based data collection protocol. Collection Tree Protocol is an address free protocol and each node selects the parent node based on the routing gradient.

The collection tree protocol uses beacons (routing messages) to construct and maintain the routing tree. The standard consists of three logical components [19]:

a. Routing engine: Routing engine decides which neighbor should be the next-hop. Each node in the network selects a parent node to forward a packet except for the root node.
b. Forwarding engine, which is responsible for forwarding the data packets. It decides if a packet is to be forwarded and when to forward it. It makes decision for packets received from other nodes as well as packets generated by node itself.
c. Estimating Engine: Estimating engine is responsible for estimating the link to the adjacent nodes, also known as neighbors. It calculates single-hop ETX values with all the neighbors.

The metric used in CTP to choose a parent node is the expected transmissions ETX [23]. The ETX of a node is equal to the ETX of the parent node plus ETX of its link to the parent.

In an area of deployment, all nodes send beacons at a fast rate to get information about the parent nodes and their quality. When a packet is received by a node, the sender is selected as a parent and quality is calculated. Parent node is chosen by selecting best node in its routing table. The parent node is changed only if the quality of new parent is

significantly better than current one. Beacons are sent in CTP based on a trickle algorithm; once the tree is stable the frequency of the beacons becomes less until there is a situation that requires resending more frequent beacons [11].

CTP uses Four-Bit estimator as its Link Quality Estimator [14]. Four-Bit gets information from three different layers. *White bit* is from physical layer which provides information that the packet has very low probability of decoding error. If *white bit* is set, it shows that the link is good and error probability is very low. *Ack bit* is from Link Layer and it shows that an acknowledgement is received for a sent packet. A set *ack bit* shows that the acknowledgement is received for sent packet. Two bits of information are provided by Network layer, *pin bit* and *compare bit*. Pin bit is used to pin the link table entry, and by doing that stopping the link estimator from removing that entry. A set pin bit shows that the entry cannot be removed from link table by link estimator. Compare bit like name shows compare the link table entries and finds if a better entry is available or not.

2.3.2 Contiki Collect

CTP has a modified version in Contiki OS named as Contiki Collect [27]. It is very similar to CTP as Contiki Collect is also a tree based protocol and uses ETX as routing metric. Unlike CTP, Contiki Collect has four components:

a. Link Estimator
b. Neighbors Manager
c. Neighbor discovery
d. Collect

Link Estimator in Contiki OS like CTP is ETX. It calculates 1-hop ETX. 1-hop ETX calculates number of retransmissions required to transmit data to sink. Neighbor Manager is responsible for parent selection. It also keeps information about the neighboring nodes which are already detected and works as a routing table. It contains each node's id and ETX value. It also has two timers. First timer is responsible to look for any entry which is not updated for too long time and second to update the link estimation bock by removing values from it. Neighbor discovery block is responsible to transmit packets to detect neighboring nodes within range. The final block is collect block which handles all transmissions and receptions of packets. It is also responsible to update other nodes if a major change occurs in current node.

2.3.3 WiseRoute

Another simple tree based collection style routing protocol used in MiXiM, a framework of OMNeT++, is WiseRoute. WiseRoute is implemented in CSEM WiseNet solution [18].

The tree starts by initiating a route flood packet from the sink to the network. When a node receives the route flood it will check the RSSI value of the incoming packet, if it is higher than a pre-set threshold it will set the node that forwarded the route flood packet as its parent node. Any duplicate route flood packet coming from any other node will be discarded. This means that at the current tree build up, the parent node will be selected based on the first received route flood packet not on the best quality path. Moreover, the link quality only considers only a 1-hop measure and not an end to end overall quality.

WiseRoute has implemented an option to choose a route with a better RSSI value instead of choosing the first best route.

WiseRoute uses RSSI (Received Signal Strength Indicator) as its Link Quality Estimator. RSSI is hardware based metric and doesn't need too much computation. RSSI is received by sampling the signal strength of the first 8 symbols after SFD (Start of Frame Delimiter) in a packet. No processing is required to get the RSSI value. It is worth mentioning that RSSI threshold above -87dbm will give more than 80% PRR. Anything lower than -87dBm RSSI, only 2dBm difference will significantly impact PRR and starts the grey area. Being hardware based metric RSSI gives a quick estimate of a link if it's in grey area.

2.3.4 Directed Diffusion

In [6] C. Intanagonwiwat et. al. proposed Directed diffusion, an example of Content based routing protocols. In directed diffusion sink node floods the network by query packet. For example the sink can request the moisture in soil of a particular location. That packet has the requested data listed so that every node is notified about the request. All nodes identify the neighbors by a unique id. All nodes are aware of application and that's why accurate data is sent to the destination. Source nodes forward data using gradients towards base station. A node maintains the gradient of the neighbor it received interest from. Delay can be the metric to choose the best path towards sink. A gradient is weighted using the amount of data that gradient traverses. A node can use multiple gradients to multiple parents to forward data towers sink. Base station selects best route and increases the weightage of routes and decreases the weight of others. Intermediate

nodes aggregate the data and that data is forwarded towards base station using gradients according to their weights. The best route is maintained by base station by resending the interests along the routes and the gradients of intermediate routes are maintained. The problem of directed diffusion is the initial route selection phase in which lots of energy is consumed and as mentioned before energy is a big constraint in wireless sensor networks and it should be conserved using protocols and software advances not the other way around.

2.3.5 LEACH

LEACH acronym for Low Energy Adaptive Clustering Hierarchy) is a routing protocol for Wireless Sensor Networks [26]. It is designed to monitor the environment remotely. It is a cluster style routing protocol where number of cluster heads collect the data and then the data is transferred to base station. LEACH provides energy efficiency by random rotation of cluster heads to conserve energy. LEACH operates in two phases:

a. Setup phase
b. Steady state phase

Like name setup phase is responsible for setting up clusters and selecting cluster heads. Cluster heads are selected using a random generated value. The cluster is selected for one round. Once CH is selected it notifies all other nodes about being CH. All nodes in select CH based on signal strengths and notify CH to get a TDMA time slot for transmission from CH. In Steady state phase all nodes send data to CH, which collects data from all nodes and transmits to Base Station for a certain time. After that time the network again goes through Setup phase and selects new Cluster Heads and creates new clusters.

2.3.6 Flooding and Gossiping

Flooding and Gossiping are one of the simplest routing strategies. Flooding has a very basic concept of receiving data from one node and transmitting it to all other nodes in vicinity. When other nodes receive that packet they do the same. Every node broadcasts every packet to all neighboring nodes. It guarantees the packet delivery to destination node. But because of each node broadcasting every packet it receives to all other nodes in range, it can overflow the network. For that reason another protocol Gossiping is used to solve the problem. Instead of broadcasting packets to all nodes in vicinity, it sends packet to a randomly chosen neighbor. By using this approach, the problem of overflowing the network is solved but it is a very slow approach and has too much propagation delay.

2.3.7 RPL

RPL is an IPv6, tree style routing protocol which transmit data from source nodes to sink nodes [25]. Several routes are possible in RPL which was designed for low power and lossy networks. RPL allows different link quality estimators to be used to select route for data transmission. It was created to give flexibility in usage and that's why multiple link estimators can be used to estimate the route for example hop-count or ETX [23]. RPL builds Directed Acyclic Graph (DAG) a topology with no cycles for multiple sinks and Destination Oriented Directed Acyclic Graph (DODAG), a topology for single root with no cycles. These topologies are built by nodes sending messages with information to discover other nodes.

2.4 Fuzzy Logic

Fuzzy Logic is a mathematical way of expressing human reasoning in mathematical notation [28]. Only two states are allowed in crisp conventional binary logic. A proposition is either true or false in classical reasoning. Fuzzy Logic is flexible and can handle imprecise data which cannot be exact but an approximation. A fuzzy set is a set without any clear distinction from other and has no sharp boundaries [28]. If-then rules are used to represent the knowledgebase. For example a fuzzy rule can be "if temperature is 'low' and water is 'cold' then 'increase' the heat". Here low cold and increase are fuzzy descriptors. A Membership function is needed to represent a fuzzy set. It gives a degree of membership to the member of sets. A membership function maps the elements into numerical values between 0 and 1. Fuzzy Logic was first introduced by Dr. Lotfi Zadeh in 1960s [29].

2.5 Determining Fuzzy Membership functions

A fuzzy set is described by its membership function. Membership function and its shape is very important and should be assigned carefully. Membership values are assigned through various methods in fuzzy logic. Sometimes it is done by intuition, sometimes by some other computational techniques. Several techniques are defined in [20]. Intuition, Inference, Rank ordering, Angular fuzzy sets, Neural networks, Genetic algorithm, Inductive reasoning are few of the techniques discussed below.

2.5.1 Intuition

With a thorough knowledge of the problem and the linguistic variable, membership values can be based on human's understanding and intelligence.

2.5.2 Inference

The knowledge to perform conclusive reasoning, the membership function is formed from the knowledge and the known

2.5.3 Rank ordering

A polling concept where preference are above pairwise comparisons based on this membership values are assigned by rank ordering process

2.5.4 Angular fuzzy sets

Sets that are defined by angles which are repeated every 2π cycles are called angular fuzzy sets. They are applied to describe the linguistic variables quantitatively known as truth-values. When the membership value '1' is true and '0' is false, anything between these two values is partially true or partially false. The linguistic values are formed to vary with θ, the angle defined on the unit circle and their membership values are on $\mu(\theta)$

2.5.5 Neural Networks

Neural Networks are used to simulate human brain functionality and the human brain concepts are used to perform computations. The data is input and then fuzzy membership function is created. The input data is selected and is divided in training and testing data sets and then the training data is used to train the network [20].

2.5.6 Genetic Algorithm

This algorithm is based on Darwin's theory of evolution based on "survival of the fittest" [20]. He also postulated that new classes of organisms came into being through a process

of reproduction, crossover and mutation. Following steps below are used to compute membership function using genetic algorithm.

i) Initially assumptions are made to define the membership functions.

ii) These membership functions are then coded as bit stings.

iii) The coded bit strings are then joined.

iv) Then fitness function is used to evaluate the fitness of membership functions.

v) And finally these membership functions are used as parameters that define the functional mapping of the system.

2.5.7 Inductive Reasoning

Inductive reasoning can also be used to define membership functions. A database is needed for the input-output relationships. This method is better suited for static data since the membership function would change with time in dynamic data. Seven steps are required to generate membership functions using inductive reasoning.

i) Establish fuzzy threshold between classes of data

ii) Determine the threshold line with entropy minimization screening method.

iii) Start segmentation process.

iv) Segmentation will result into two classes.

v) The two classes are further partitioned so there are three different classes.

vi) This partitioning is repeated with threshold value calculations until dataset is divided into a number of fuzzy sets.

vii) Then membership function is determined based on shapes.

Chapter 3 Determination of Accurate Fuzzy Membership function for F-LQE

This chapter focuses on the determining of an accurate fuzzy set for F-LQE [15]. In order to perform this optimization F-LQE was implemented in WiseRoute and the details of this implementation is presented in the next chapter. Here in this chapter the focus is on determining a refined fuzzy set for F-LQE.

A strategy was adopted to determine the refined fuzzy set, which is explained in the following steps. First a tree structure of links is generated and a source and a sink node are selected.

1. All the parameters of all the fuzzy set are fixed for now to easily select the initial fuzzy set thresholds.
2. After that the fuzzy link quality,$FLQE$ was calculated for all the neighbors between the source node and the sink node.
3. After the fuzzy link quality between two nodes is calculated, the path quality, $PFLQE_k$ is calculated at source node. The path with best quality, $PFLQE_k$ is selected as the best path in terms of quality of the link.
4. Using the best path, multiple packets are sent from the source node to the sink node and PRR is calculated.
5. This process is repeated several times from step 2 until a refined threshold is found for the fuzzy set.

6. The threshold using which the best path (whose PRR is highest), is selected is chosen as the refined threshold.

First the MATLAB was used for verification of the proposed approach. Then that approach was replicated in WiseRoute using fixed SNR values in order to confirm its feasibility in a converge cast routing algorithm and finally the algorithm was verified using the simulation in WiseRoute with an actual channel model as opposed to simply using fixed SNR values.

A number of techniques were explained in the previous chapter for the determination of fuzzy set membership functions. The approach used is very closely related to the inference-based approach presented in section 2.5.2. Based on the results of the simulations conducted, a membership function is determined.

First a mathematical evaluation was conducted and it was seen that a slight change in the thresholds of membership functions caused the change in route selection. Each time if the membership function is varied enough that the overall quality of the link changes, the route for packet transmission was changed too. After the evaluation a simulation was performed exactly the same as the mathematical evaluation and it was again seen that the change in membership function according to link quality, also changes the next hop selection. Finally the exact strategy was applied to a real simulation scenario and after a number of testing and threshold selection in different scenarios the final threshold was selected which performed better than all other scenarios in terms of PRR and other measures.

The best fuzzy set in F-LQE was determined by the relationship of Signal to Noise Ratio (SNR) to Packet Reception Ratio (PRR) [15]. According to their observation when Average signal to noise ratio (ASNR) curve is above a certain threshold, PRR is higher than 95% while if it was less than another threshold, it was less than 25%. This observation, the PRR/SNR curve motivated their fuzzy membership function threshold. μASNR is the degree of membership which ranges between 0 and 1 based on Average SNR values. Figure 3.1 shows the membership function proposed in F-LQE by N. Baccour et. al.

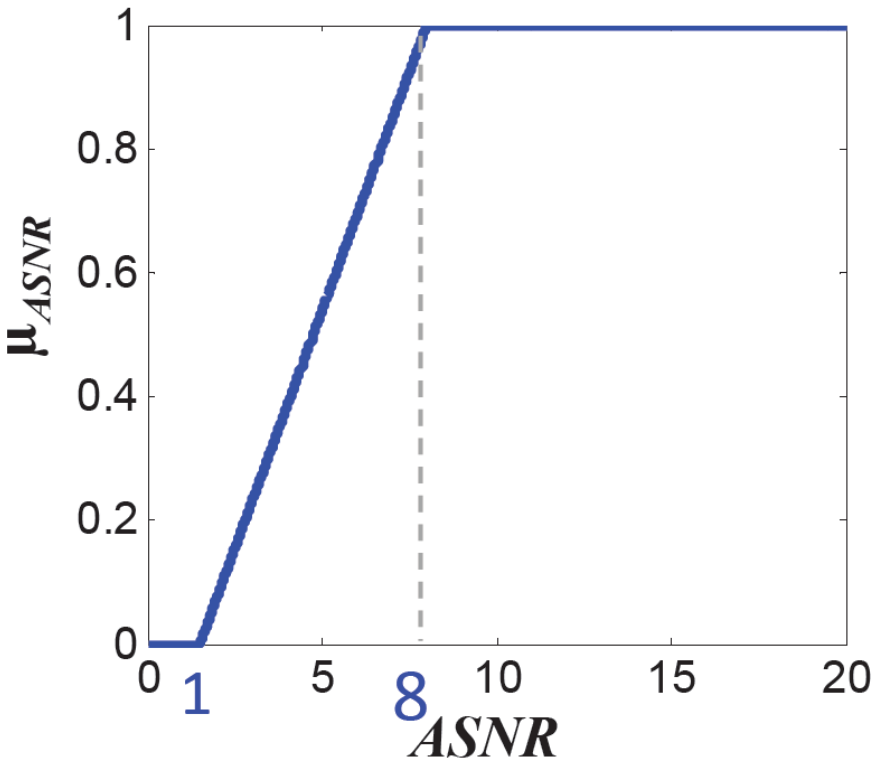

Figure 3.1: Proposed ASNR membership function in F-LQE [15]

The proposed strategy keeps the same shape of the membership functions of F-LQE. The upper bound threshold will be chosen based on the fixed parameters assigned to each

node. The lower bound threshold will be kept between 0 and 1 because that threshold if chosen even a bit bigger might just discard the link. This is because that point starts the initial value of Membership function. All the values preceding the lower bound value are discarded.

3.1 MATLAB tests

MATLAB provides a fuzzy logic toolbox which can be used to analyze and design fuzzy logic based systems. One can define own rules using AND, OR and NOT logical operators.

In order to get a preliminary confidence that a refined value of the upper bound of the membership function existed, the implementation of the algorithm was programmed in MATLAB. For evaluation a scenario was made with three paths and each path had three different SNR values.

Those SNR values were fixed in simulation time. Figure 3.2 shows the paths with SNR values fixed between them.

To calculate the path quality for the selection of the best path, the following routing metric was used to calculate the path quality and then select the best available path for data transmission.

Path quality ($PFLQE_k$) can be estimated by the following formula:

$$PFLQE_k = \beta\left(\frac{1}{N_k}\sum_{j=1}^{N_k} FLQE_{kj}\right) + (1-\beta)\min_{\mathbf{1}\leq \boldsymbol{j} \leq N_k} FLQE_{kj}$$

Where $FLQE_{kj}$ is the link quality of the j-th link on the k-th path, and β =0.7.

Where $\left(\frac{1}{N_k}\sum_{j=1}^{N_k} FLQE_{kj}\right)$ is the average path quality, which is calculated by sum of all the links in the path and divided by the path hop count. This metric considers path quality and weakest link quality, and both of these qualities are combined using weighted sum and the path with highest $PFLQE_k$ is selected. Higher value for β decreases the weightage of the weakest link.

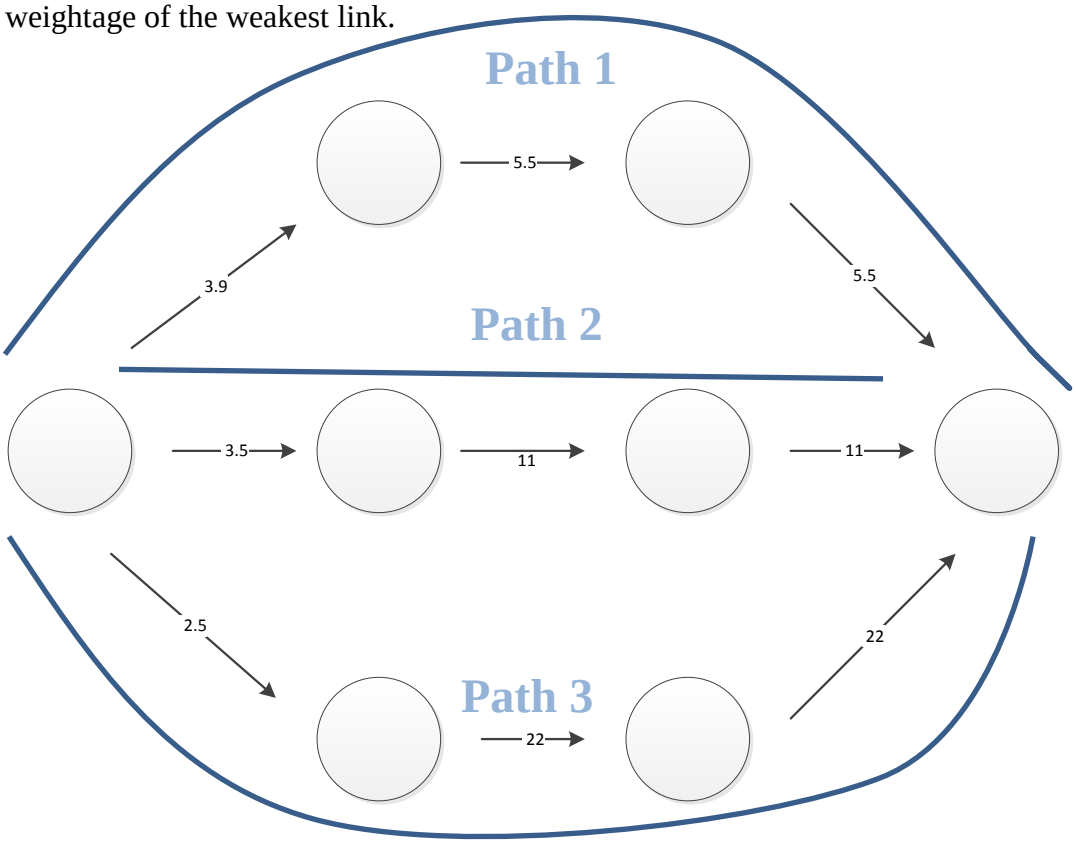

Figure 3.2: Membership function calculation using MATLAB

The membership function's lower bound started at 1 and four different upper bound thresholds 5.5, 11.0, 16.5, 22.0 were selected. When the tests were run with these four different upper bound thresholds for the SNR membership functions, different routes

were selected. It was concluded that improper thresholds would lead to choosing a path which is worse than best quality path. The membership function from 1 to 11.0 selected one path while other three membership functions selected other paths which were not the best quality paths. Figure 3.3 shows the membership functions selected for the mathematical evaluation.

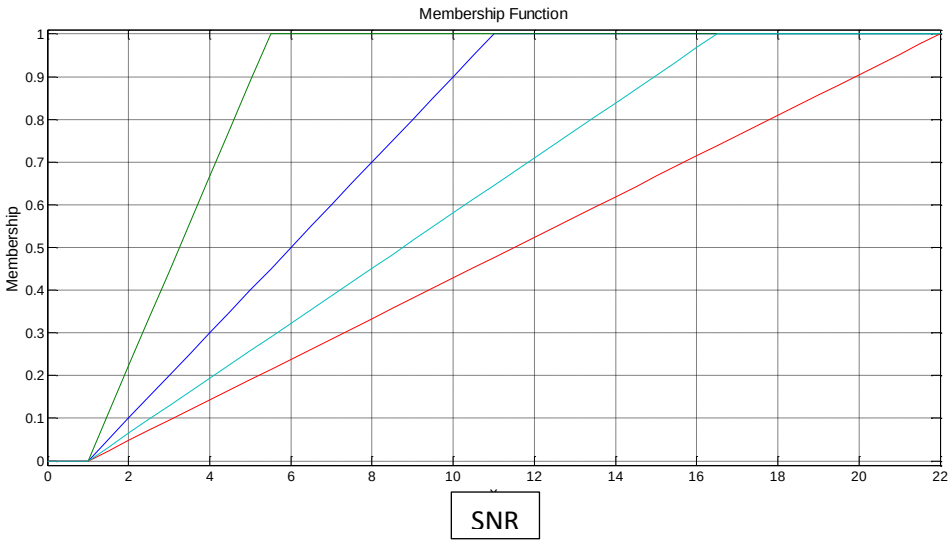

Figure 3.3 Membership functions

The path kept changing with different thresholds. For different upper bound thresholds different paths were chosen.

It was concluded that the membership function can affect the route selection and with that if the membership functions are not selected carefully the resultant route will not be the best among all possible routes and this can result in packet loss or less stable topology.

3.2 WiseRoute Implementation of the Optimization Algorithm

The evaluation was done mathematically in previous section. This evaluation performed in previous section also required testing in routing protocol. WiseRoute is chosen as the testing protocol and Fuzzy Logic based link quality estimator, F-LQE, was implemented into it. WiseRoute [18] a simple loop-free tree based routing algorithm which was implemented in CSEM WiseNet solution. The implementation of F-LQE in WiseRoute is available in Chapter 4. WiseRoute is chosen due to its simple architecture and its availability in MiXiM [17], an OMNeT++ [16] based modeling framework for mobile and fixed wireless sensor networks. It has a number of mobility models; detailed radio models for interference estimation, power consumption by radio transceiver and also has wireless MAC protocols implemented into it.

The Fuzzy set determination was performed in two steps in WiseRoute;

- In the first step the membership functions and routes were tested as used in MATLAB without a channel model and fixed SNR values were used on the nodes and the scenario was replicated with a routing protocol
- In the second step WiseRoute implementation was used with SNR values determined by thermal noise by each node to make more realistic scenario.

3.2.1 Replicating MATLAB evaluation of membership function in WiseRoute with fixed SNR.

In the first simulation sensor nodes were placed in a way that only three possible routes are possible for data transmission. This was done by placing only strategic nodes in each other path Figure 3.4 shows the network setup in WiseRoute to replicate the MATLAB scenario. Due to range of nodes, all nodes were placed far away in x, y coordinates from each other to avoid any extra route creation for data transmission and create only three paths for data transmission. SNR is used in this section for the link estimation and only one membership function Average Signal to Noise Ratio (ASNR) is used.

This setup is slightly different from the previous MATLAB setup because, in MATLAB setup there were only three routes and if same setup is replicated in WiseRoute, other direct links are created to transmit data due to nodes range problems. The slightly different setup from WiseRoute was also tried in MATLAB and the results in this setup were also similar to the ones in MATLAB

This simulation also showed that different upper bound thresholds for SNR membership functions select different routes for data transmission. The routes were created based on the fixed SNR values. WiseRoute code was modified for manual assignment of SNR values and this assignment of fixed SNR was done using Node id's and different condition statements. The values for SNR were assigned manually for each node and based on different membership function thresholds, different routes were selected. This again validated the previous study that the route can be selected based on the membership function threshold and wrong selection of membership function could lead to wrong route

selection. When similar scenario was implemented in MATLAB, the thresholds were same. The previous scenario cannot be replicated due to the range reasons explained above but the scenario similar to current one showed exact results similar to this setup.

Thresholds Upper bounds	3.5	5.5	11
Path1			
Path 2			
Path 3			

First threshold of 5.5 selected the path 1, the upper path, for data transmission. 3.5 thresholds chose the center path, i.e. path 2, for data transmission and 16 selected the lower path, path 3. These thresholds were same as another MATLAB test showed.

Next the Fuzzy logic based WiseRoute was implemented with full channel model and the effect of membership function was seen on route selection.

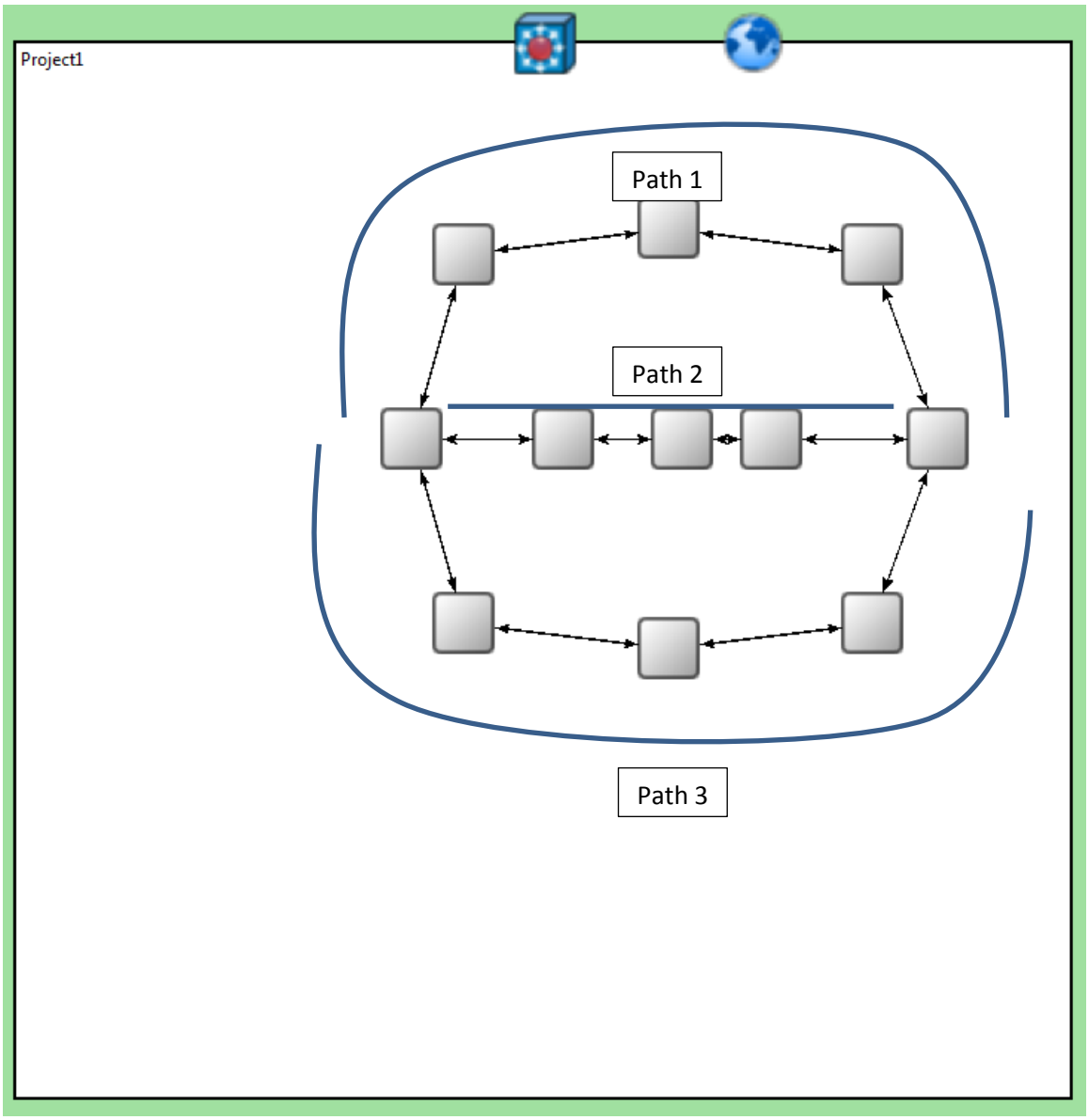

Figure 3.4: WiseRoute simulation to replicate MATLAB mathematical evaluation

3.2.2 Replicating MATLAB evaluation of membership function in WiseRoute with original channel and SNR based on Thermal noise.

This section discusses the membership function selection process evaluation for fuzzy link quality estimator implementation in WiseRoute. Here the WiseRoute simulation in MiXiM was performed with fuzzy link quality estimator with SNR link quality and only one membership function was used for evaluation. The entire simulation scenario was kept exactly the same as the previous simulation. This simulation was different from the previous simulation in a way that the SNR values were assigned by the simulator based on the thermal noise Random thermal noise was assigned to each node ranging between -85.9dBm to -99.2dBm. Based on the thermal noise and signal strength, the Signal to Noise Ratio (SNR) was calculated. Figure 3.5 shows the WiseRoute simulation with original channel model.

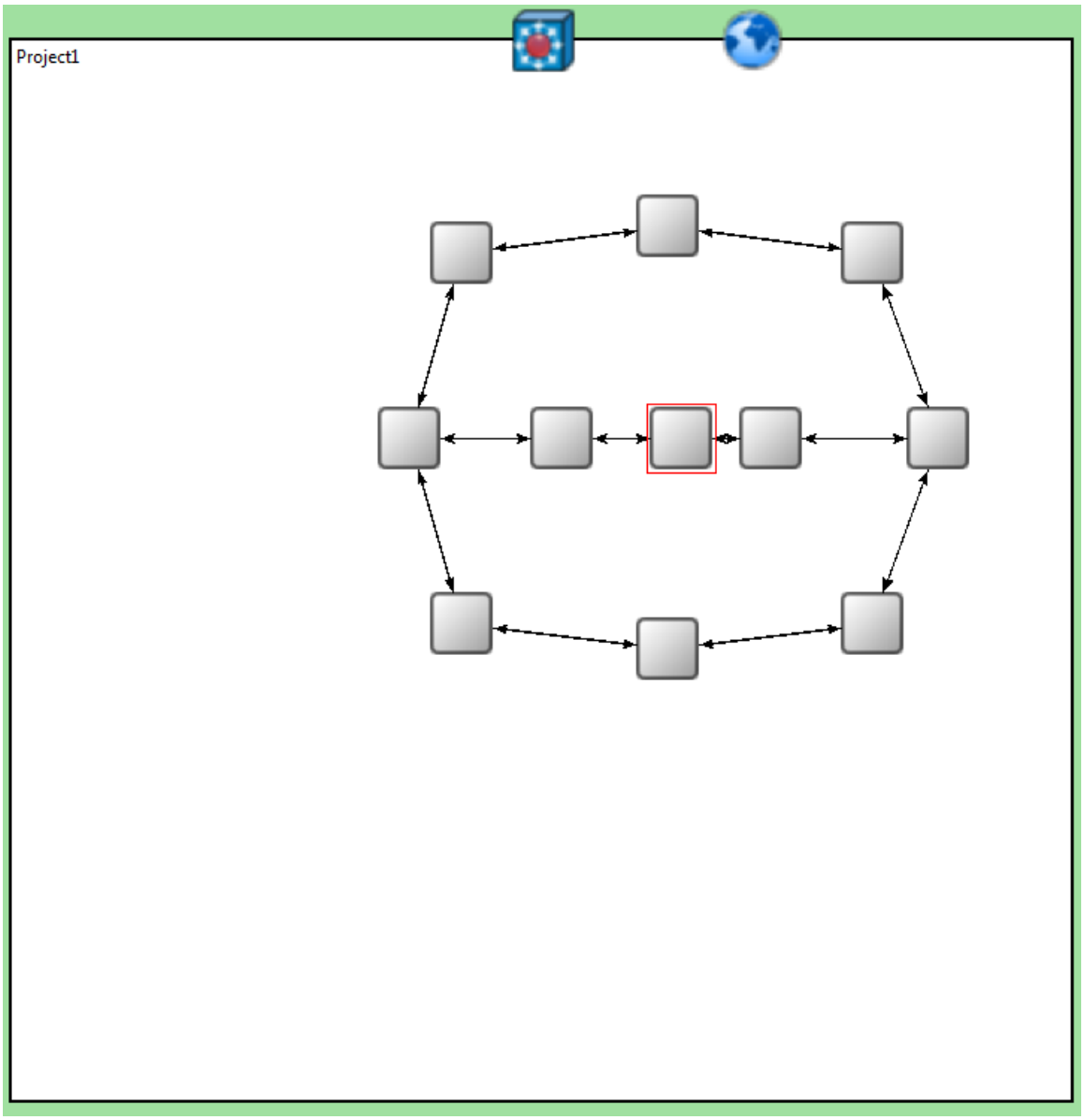

Figure 3.5: WiseRoute simulation with SNR assigned by thermal noise

This simulation with WiseRoute and fuzzy link estimator also showed that different membership functions give different routes for data transmission based on the SNR of the link.

The above experiments showed us that the membership function determination is an important step in fuzzy logic based link quality estimators. Although intuition and other

methods were recommended and used by previous studies but intuition or random experiments without having the knowledge of link estimator functionality will end up in wrong route selection.

These experiments above were performed basically to determine that the upper bound threshold and the shape of a fuzzy set. This was done to show that these two fundamental components of a fuzzy logic based link quality estimator will influence the path selection for transmission of data.

Chapter 4 F-LQE implementation in WiseRoute

This chapter explains the WiseRoute routing protocol functionality and how the RSSI based link estimation works. After that it explains a recently proposed link estimator F-LQE, and its implementation into CTP [15]. Then it also has the explanation of my implementation of F-LQE into WiseRoute.

According to N. Baccour et al. in [15] the quality of a link is combination of several channel properties and they proposed F-LQE a fuzzy logic based link estimator. It considers four link properties, i.e. Packet Delivery, Asymmetry level, Stability Factor, and Channel Quality. These four properties are then fuzzified using fuzzy logic and then link is estimated.

WiseRoute is a simple converge cast based routing protocol that has been implemented in MiXiM [17] a framework on OMNeT++ [16]. WiseRoute is a very efficient protocol in terms of packet reception ratio and energy consumption [18]. WiseRoute uses RSSI as a link estimator and if a route flood packet is received with RSSI above a certain predefined threshold, the node is selected as a parent node and cannot be changed until a new routeflood packet is generated by the root node. If a route flood packet is received with an RSSI threshold lower than the predefined threshold then no parent node is recorded for that node and any data packet received by that node is flooded to the network with the hope that the packet will reach the sink node.

In this thesis the Fuzzy Logic based link Quality Estimator was implemented in WiseRoute. Some modifications were done in WiseRoute so that the flooding mechanism of WiseRoute is removed. A number of routefloods were transmitted in the beginning of

the network before data transmission. Using those routefloods PRR was calculated and the link metrics based on PRR were calculated. This thesis also added the SNR calculation module in WiseRoute to calculate the SNR for each packet and calculate the four link properties of F-LQE.

4.1 WiseRoute

WiseRoute is a simple converge cast based routing protocol implemented in MiXiM. It was implemented in CSEM WiseNet solution [18].

In WiseRoute a tree generation is started by the root node generating a route flood packet. Every node which receives a route flood packet checks for the RSSI value of the packet it received. If the RSSI value is higher than a predefined threshold, the sender node will be selected as a parent node for data packet transmission and that routeflood packet will be further transmitted. If a node received a duplicate packet, it will be discarded. The duplicate packets will be discarded even if the recently received packet has higher RSSI value than the previous one which was used to select parent node. So any packet which is received first will be used to select parent node not the packet with best RSSI value. Also the link considers only 1-hop quality of the link and not the overall link quality.

WiseRoute includes following features:

1 Route flood initiation: The Sink node will initiate a route flood packet that will flood the network. This flooding packet can be initiated periodically depending on the application or the given scenario.

2 Handling Application layer messages: During simulations the sensor readings are sent down from the application layer to be forwarded to the sink, there are two options for these packets:

   a. The application layer sends out a broadcast flood data packet if it didn't receive any route flood packet from the sink and hence doesn't have any parent node, specifying the sink as the final destination.

   b. The node received a route flood packet from the sink and knows its parent node. The packet will be sent out as a pure unicast to the parent node.

3 Handling of incoming packets: If the message comes from the lower layer there are different options:

   a. It is a route flood packet sent out from the sink.

      i. If it is first time received, its information will be saved. (Initial source, source, BER, RSSI) and then the source and RSSI and BER will be replaced and forwarded to the next hop.

      ii. If it is a duplicate, then it will be discarded.

   b. It is a data packet.

      i. If it is a flood packet sent out from a node that didn't have a route to the sink then it will be forwarded as a flood, or discarded if it is a duplicate.

      ii. If it is a unicast packet forwarded by another node:

1. If the current node is included in the tree then it will forward it to its parent node as a unicast packet.
2. If the current node never received a tree flood from the sink, then the next hop address will be replaced by the sink address and forwarded as a unicast packet, if the sink is not one hop away then packet will be discarded.

4.2 F-LQE

F-LQE is a recently proposed fuzzy logic based link quality estimator. It considers four link properties and combines them using EWMA filter. F-LQE considers Packet Delivery, Asymmetry level, Channel Quality, and Stability factor.

Packet Delivery is calculated using Smoothened Packet Reception Ratio (SPRR) that is the same as WMEWMA [24]. F-LQE uses EWMA filter [24] to smoothen the Packet Reception Ratio to calculate SPRR.

$$PRR = \frac{Number\ of\ Received\ Packets}{Number\ of\ Sent\ Packets}$$

$$SPRR(\alpha, \omega) = \alpha \times SPRR + (1 - \alpha) \times PRR$$

Where α is a smoothing factor and $\alpha \in$ [0..1] while ω is estimation window.

Asymmetry level is computed by computing the difference of connectivity between forward links to backward link in the ω window. It is calculated to know if a packet can be acknowledged or not.

$$ASL(\omega) = |PRR_{forward} - PRR_{backward}|$$

Channel Quality is assessed using SNR hardware based metric. It is computed by subtracting noise from the received signal.

The Stability factor is calculated by dividing the standard deviation of PRRs by the mean of PRRs.

$$SF = \frac{\sigma_{PRR}}{\mu_{PRR}}$$

Where σ_{PRR} is the standard deviation of n PRRs and μ_{PRR} is mean of n PRRs.

The fuzzy rule to combine these properties is

"**IF** the link has high *packet delivery* AND low *asymmetry* AND high *stability* AND high *channel quality* **THEN** it has high quality."

And this rule is translated into a fuzzy equation as

$$\mu(i) = \beta.\min\left(\mu_{SPRR}(i), \mu_{ASL}(i), \mu_{SF}(i), \mu_{ASNR}(i)\right) + (1-\beta).mean(\mu_{SPRR}(i), \mu_{ASL}(i), \mu_{SF}(i), \mu_{ASNR}(i))$$

Where $\mu(i)$ is the membership in fuzzy subset for high quality link. β is constant in [0…1] and $\mu_{SPRR}(i), \mu_{ASL}(i), \mu_{SF}(i), \mu_{ASNR}(i)$ are fuzzy variables.

$$LQ(\omega) = 100.\mu(i)$$

The above fuzzy equation is then smoothened using EWMA filter to get the final link quality estimate.

$$FLQE(\alpha, \omega) = \alpha \times FLQE + (1 - \alpha) \times LQ$$

Where α is a constant and $\alpha \in$ [0..1] and ω is the estimation window [15].

Figure 4.1 shows the membership functions for all four link properties used in F-LQE.

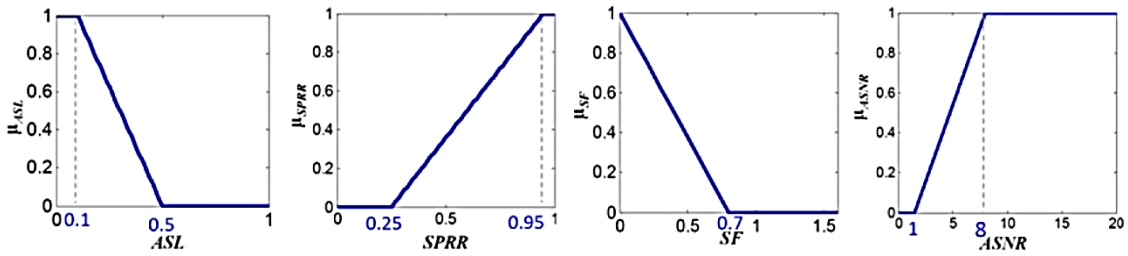

Figure 4.1: Membership functions [15]

4.3 Modifications to WiseRoute to Accommodate F-LQE

In this approach I implemented F-LQE in WiseRoute and a number of modifications were made in WiseRoute in the process of implementing F-LQE. The modified WiseRoute code is available in Appendix B. Major modifications in WiseRoute are briefly explained below.

1) I added an acknowledgement mechanism which enabled the calculation of the packet reception ratio which works only in setup phase. In data delivery phase the ack mechanism is turned off.
2) To replicate the MATLAB evaluation methodology, I implemented fuzzy link estimator based on one membership function, i.e. Channel Quality. For the first scenario all values of SNR were manually assigned to each node and for the

second full channel model was used with SNR values assigned in the simulation based on thermal noise.

3) The hop-count feature was added which considers the number of hops used by packet to reach sink node. Each packet keeps track of the number of hops it used to reach destination.
4) Instead of using one route flood in the beginning and recreating tree while data transmission if the link broke, Routeflood was retransmitted a number of times in the beginning of the network to calculate PRR between nodes.
5) For SNR calculation, several modifications were made in Mac layer and Network layer. The SNR part was added in CSMA802154 to calculate the SNR values and then forward the values to WiseRoute.
6) The broadcast feature in WiseRoute was stopped so that the packet is transmitted through the routing tree. It is not appropriate to flood the packets if a tree is not selected while testing the proposed approach. Although this feature can be turned on according to user needs.
7) The fuzzification modules for four link properties of F-LQE are implemented. The fuzzification modules are implemented for SPRR, ASL, SF and ASNR.
8) SPRR or Smoothened Packet Reception Ratio is calculated by PRR and then the values are fuzzified using the fuzzy module implemented in WiseRoute.
    a. Those values are then smoothened using EWMA filter.
9) Asymmetry level or ASL is calculated by calculating the asymmetry between uplink and downlink. Asymmetry is calculated using bidirectional PRR. $PRR_{Forward}$ and $PRR_{reverse}$.

a. For $PRR_{reverse}$ I used the Acknowledgements I added for all the routeflood packets sent.
b. The ASL values are then sent to the fuzzy module to fuzzify the ASL values and get the fuzzy value.

10) For Stability Factor SF, I calculate the PRR for a window of 5 packets and for that reason PRR for five packets was calculated and then mean, variance and standard deviation of PRR was calculated.
    a. For stability factor standard deviation of PRR was divided by mean of PRR.
    b. The Stability factor is then fuzzified using the implemented fuzzy logic module for stability factor.
11) SNR module is added in the CSMA802154 and WiseRoute is calling the method from there and using the values.
    a. The SNR values are fuzzified using the implemented method for SNR.
12) After the calculation of fuzzy values for all the metrics, the values are then combined using fuzzy rule implemented in [15]. The mean and minimum F-LQE values were also used in the proposed formula.

$$\mu(i) = \beta.\min\left(\mu_{SPRR}(i), \mu_{ASL}(i), \mu_{SF}(i), \mu_{ASNR}(i)\right) + (1-\beta).mean\left(\mu_{SPRR}(i), \mu_{ASL}(i), \mu_{SF}(i), \mu_{ASNR}(i)\right)$$

13) The route selection and update mechanism is also updated by calculating the fuzzy metric in route update method. Here a module was to calculate a routing metric based on F-LQE.
14) Finally the parent node is selected based on the best route calculated.

Chapter 5 Analysis of Refined Fuzzy Sets

This chapter discusses the results of the simulations that were performed in WiseRoute using the fuzzy link estimation. Section 5.1 will explain the determination of refined fuzzy sets calculated from the proposed approach explained in chapter 3. Then, later sections explain the simulation scenarios in detail. Then the next section will discuss the results.

Parameters	Value	Units
Topology		
C_x	150	meters
C_y	150	meters
N_{nodes}	30	-
X_{so},Y_{so}	0,0	Sink coordinates
X_{sk},Y_{sk}	150,150	-
General		
Nb. of Data Packets	150	-
Interval	7.5	Seconds
Thermal Noise	-85.9 - -99.2	dBm
Mac Layer	802.15.4	-

Table 1: Simulation Parameters

In all simulation scenarios, the field size was 150m x 150m. 30 static nodes were used in all simulations, arranged in a grid. Figure 5.2 shows the grid setup of the network. This

setup was chosen because this work is based on converge cast based protocols which has number of routes from source to sink so it was required to have a number of routes from source to sink. In this setup, the source and sink were in two opposite ends of the network and a packet had to travel through a number of hops to reach sink node. In periodic manner 150 packets were sent from source node 29 to sink node 0 after an interval of 7.5 seconds. Random thermal noise was assigned to each node ranging between -85.9dBm to -99.2dBm. MAC layer for all simulations is based on 802.15.4. Except for scenario 5 and scenario 6 discussed below routing tree was not recreated once the topology was setup. This is due to the fact that the network is static and no sudden breakage of link is expected.

5.1 Determining an refined fuzzy set

Before the final simulation for the comparison of different WiseRoute scenarios, best fuzzy set is determined using the proposed approach from chapter 3. In this section several fuzzy set thresholds were tested for the simulation scenario explained below. The lower bound threshold of fuzzy membership function was fixed at 0 due to the fact that if the lower bound threshold is set at a higher value, few fuzzy sets will be completely discarded. The upper bound threshold was varied from 1 to number of values. This process was done to find out an best fuzzy set threshold which chooses the best path with highest PRR. In Figure 5.1 a comparison was given for number of upper-bound thresholds of a fuzzy set and their packet reception ratio was also given and it was shown that different upper bound thresholds have different PRRs.

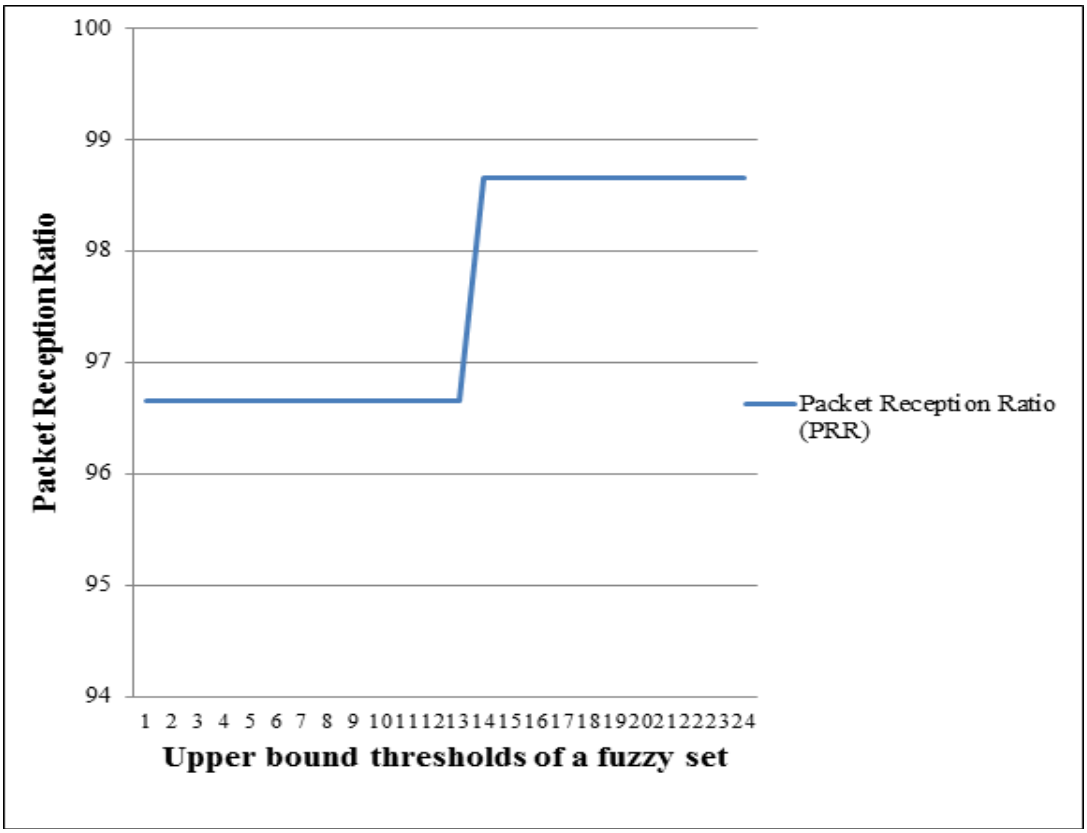

Figure 5.1: PRR calculated in different paths using number of upper bound thresholds of fuzzy set

It was concluded that from threshold 0 to 13 the path remained same for all thresholds with PRR of 96.66%. Once the threshold was greater than 14 a different path was chosen which gave much better PRR than the path chosen by upper bound threshold between 0 and 13. The path chosen by upper bound threshold greater than 14 gave PRR 98.66% which was much higher than previous path. So two fuzzy sets were found one with lower bound set to 0 and upper bound threshold ranging from 0 to 13 and second with lower bound set to 0 and upper bound ranging from 14 and onwards.

Next the fuzzy sets determined in previous section are compared to number of WiseRoute modifications and original WiseRoute.

Following were the simulation scenarios compared.

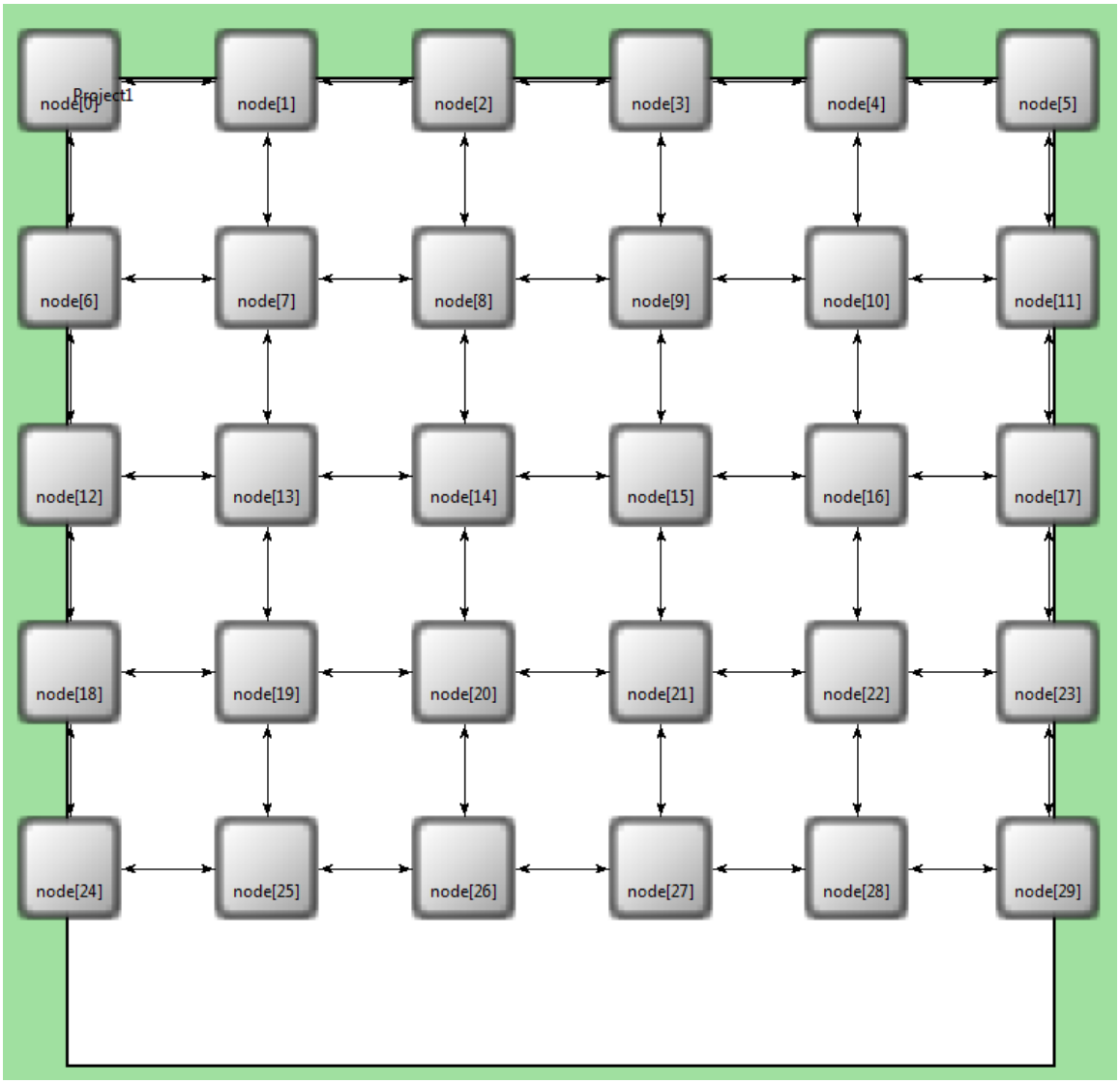

Figure 5.2: The network setup.

The image above shows the network setup in which the adjacent nodes are connected to each other. Each node is connected to a maximum of 4 nodes which are next to it.

5.2 Scenario 1: SNR based WiseRoute (Threshold less than refined threshold 14)

In this scenario, SNR based fuzzy link estimator was implemented in WiseRoute and the fuzzy membership function ranged from 0 as lower bound threshold and 0-13 as upper bound thresholds. Upper bound threshold of 12 was chosen as mentioned in previous section all values between 0 and 13 as upper bound threshold gave similar results.

5.3 Scenario 2: SNR based WiseRoute (Refined threshold)

This scenario is same as previous one where SNR based fuzzy link estimator is implemented in WiseRoute but the membership function for SNR was ranging from 0 as lower bound threshold and 14 onwards as upper bound threshold. Upper bound threshold of 14 was chosen as mentioned in previous section all values from 14 and onwards as upper bound threshold gave similar results.

5.4 Scenario 3: WiseRoute with F-LQE metrics

For this scenario, Fuzzy Link Quality Estimator (F-LQE) was implemented in WiseRoute and tested with all link quality metrics used. The membership function values for all the link properties including SNR were kept exactly the same as proposed by N. Baccour et. al. in [15].

5.5 Scenario 4: WiseRoute with FLQE metrics and refined Channel Quality fuzzy set.

In this scenario, Fuzzy Link Quality Estimator (F-LQE) was implemented, with all link properties SPRR, SF, ASL were exactly the same as proposed by N. Baccour et. al. in [15] except for channel quality. In this scenario the fuzzy membership function for SNR ranged from 0 to more than 14.

5.6 Scenario 5: WiseRoute (RouteFloodInterval=20s)

This scenario is the original WiseRoute implementation. All WiseRoute parameters were used exactly as implemented in MiXiM. The RouteFloodInterval was set to 20s in this scenario.

5.7 Scenario 6: WiseRoute (RouteFloodInterval=1200s)

This scenario is exactly same as previous one except for RouteFloodInterval. All WiseRoute parameters were used exactly as implemented in MiXiM. The RouteFloodInterval was set to 1200s in this scenario.

Following metrics are used for the evaluation of different fuzzy link estimation based WiseRoute.

1. *Packet Reception Ratio*: Packet Reception Ratio is the ratio between the number of data packets sent to the number of successfully received packets.
2. *Mean Number of Hops*: Mean Number of Hops is the number of Hops a packet takes to transmit a data packet from source node to sink node

3. *Latency*: Latency is the time taken by a packet to reach sink node from source node.

5.8 Ideal paths chosen by different scenarios

This section shows the ideal paths chosen by different simulation scenarios

5.8.1 Scenario 1:

The ideal path chosen by the network in Scenario 1 was

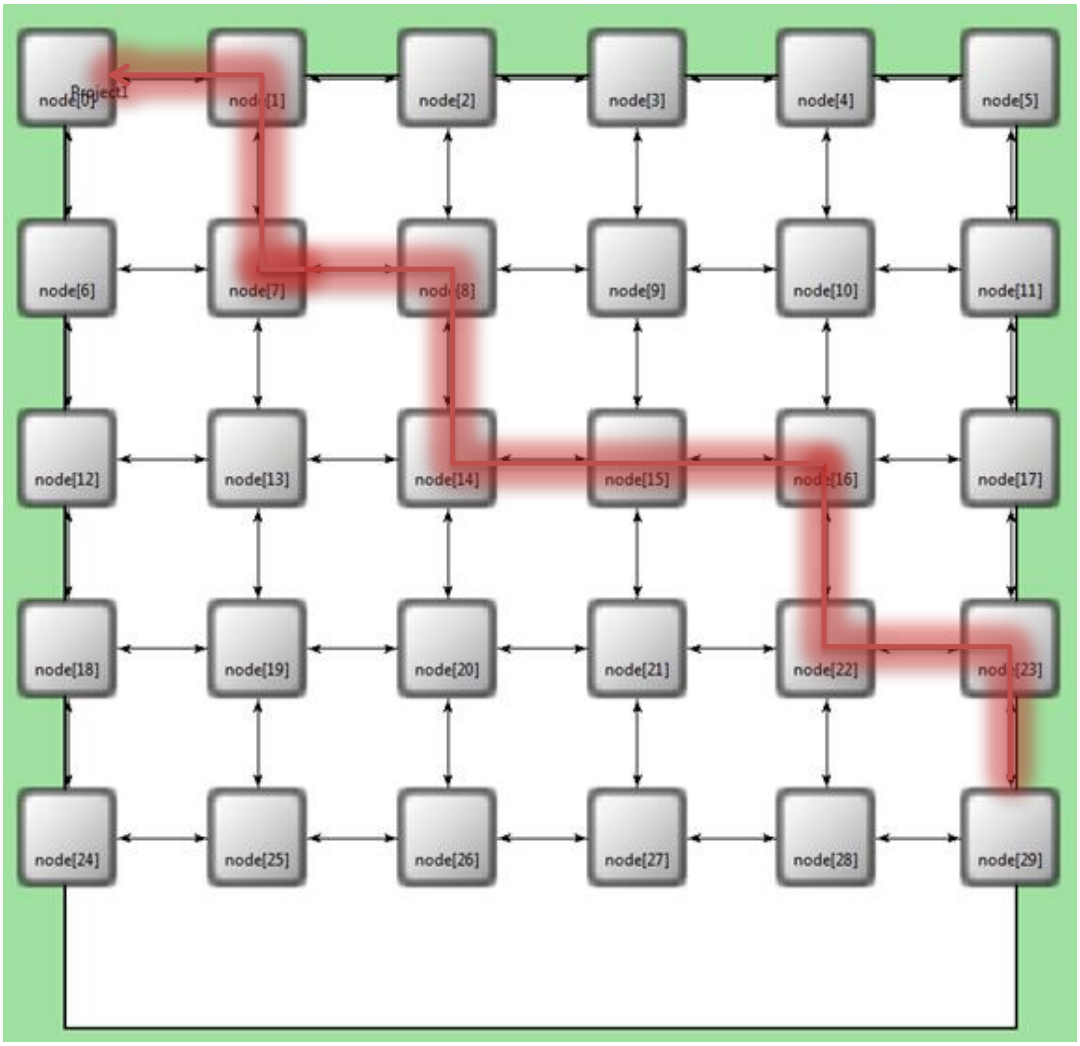

Figure 5.3: Ideal Path chosen in Scenario 1

5.8.2 Scenario 2:

The ideal path chosen by Scenario 2 was

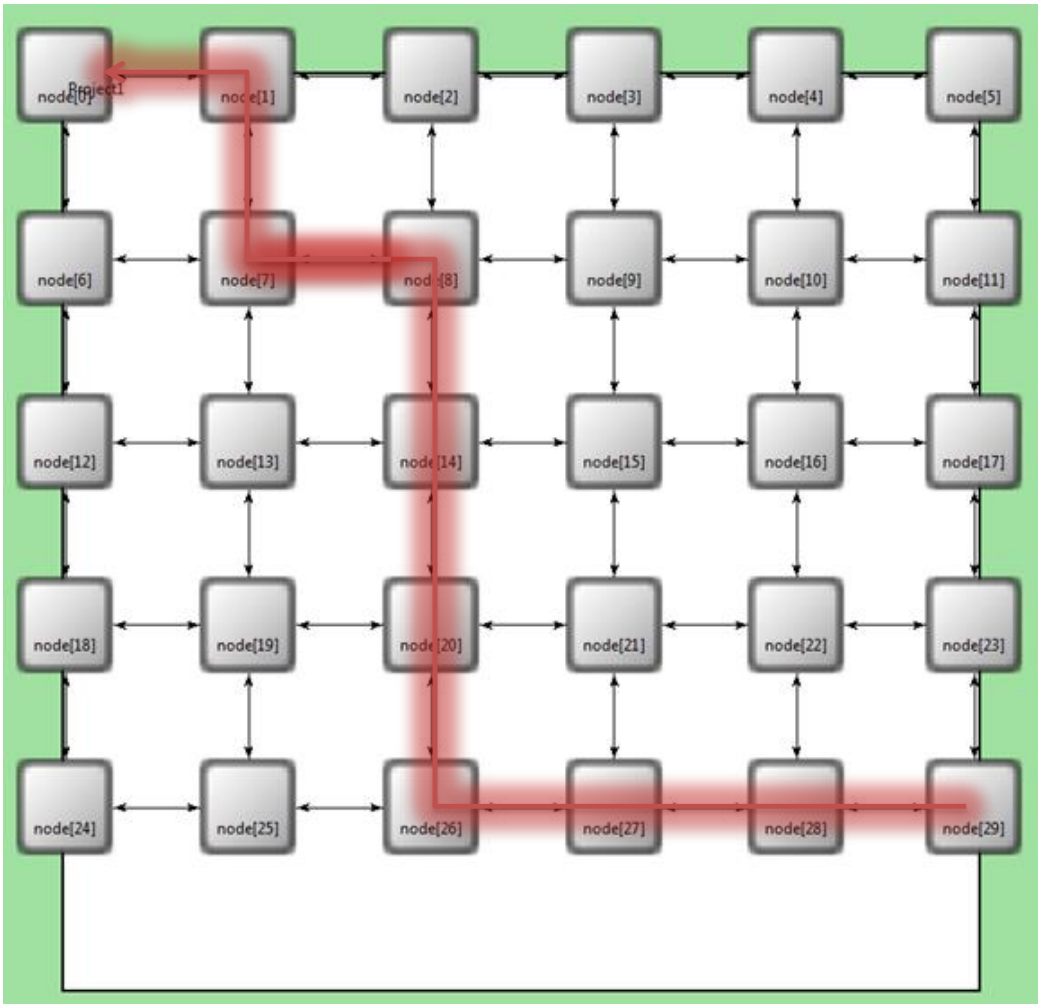

Figure 5.4: Ideal path chosen by Scenario 2

5.8.3 Scenario 3 & Scenario 4:

Scenario 3 and Scenario 4 both had same ideal paths for data transmission.

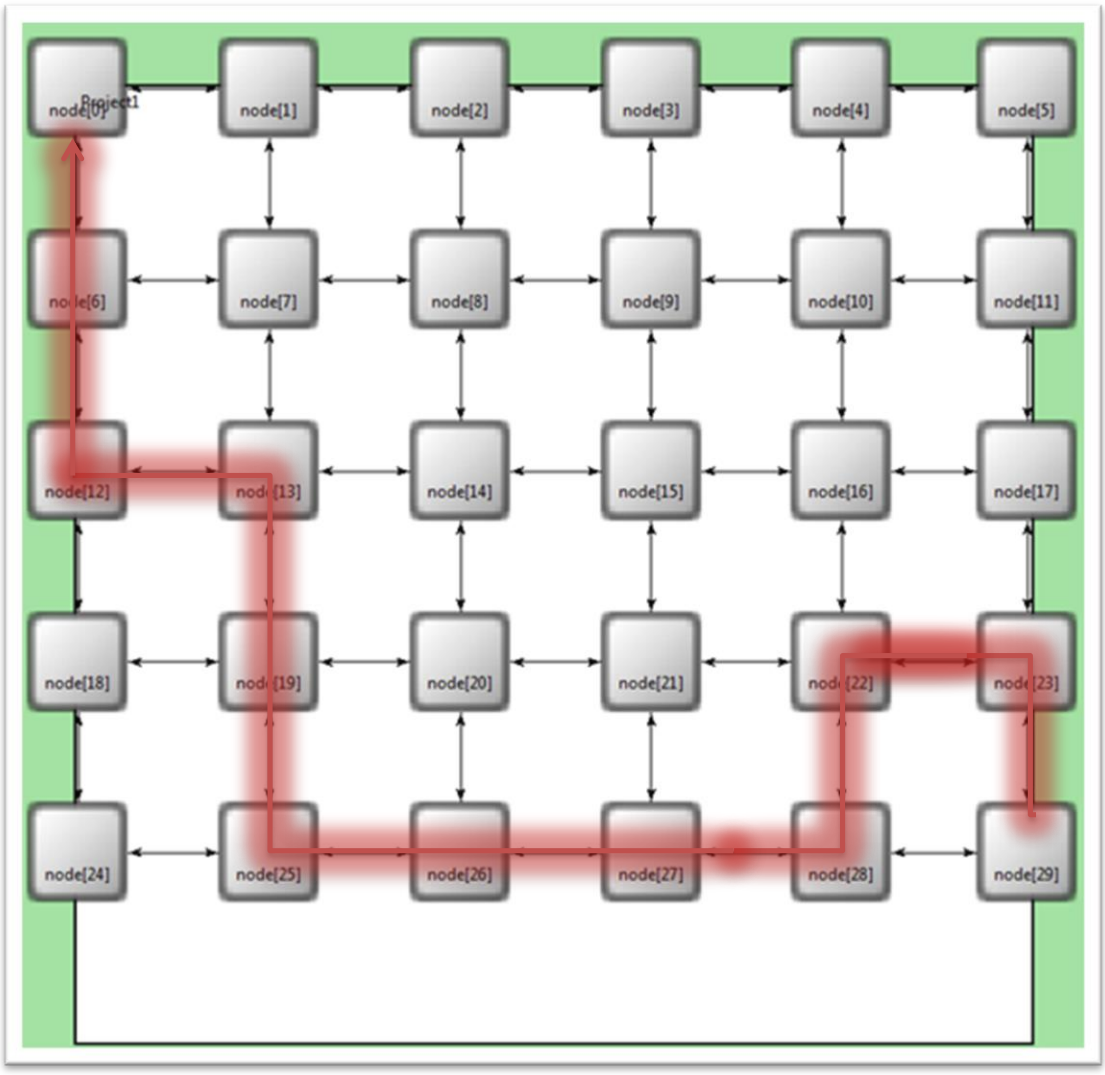

Figure 5.5: Ideal path chosen by Scenario 3 and Scenario 4

5.8.4 Scenario 5 & Scenario 6:

Different ideal paths were chosen by Scenario 5 and Scenario 6. This is due to the fact that Original WiseRoute protocol, which recreated the routing tree after a manually specified time (RouteFloodsInterval). Once the clock triggered the time specified in RouteFloodsInterval, a RouteFloodPacket was sent and a new ideal path was recreated.

Due to this too many routes were created during simulation time. Because of the reasons explained above no figure of routing paths of Scenario 5 and Scenario 6 was added.

5.9 Results and Analysis

First set of experiments will consider the packet reception ratios in different scenarios. Figure 5.6 shows the difference of packet reception ratios of all the network setups explained above. It can be seen that the optimized scenario, Scenario 2 performed better than all the other scenarios. We call it F-LQE-Opt. Several tests were performed and the results with fuzzy upper bound threshold lower than 14 in scenarios 1 performed much lower than F-LQE-Opt. Scenario 3 with F-LQE metrics performed worse than all other scenarios. Even when Channel Quality Fuzzy set was replaced with F-LQE-Opt in simulation 4 the results of simulation 3 and simulation 4 remained same. This is basically due to all the membership functions were given equal weightage and one different fuzzy set still didn't affect the overall route quality and same route was selected in scenario 4 as in scenario 3. Finally two original implementations of WiseRoute from MiXiM were simulated in scenario 5 and scenario 6. Scenario 5 and scenario 6 also performed better than scenario 3 and 4 but the PRR was not as good as F-LQE-Opt.

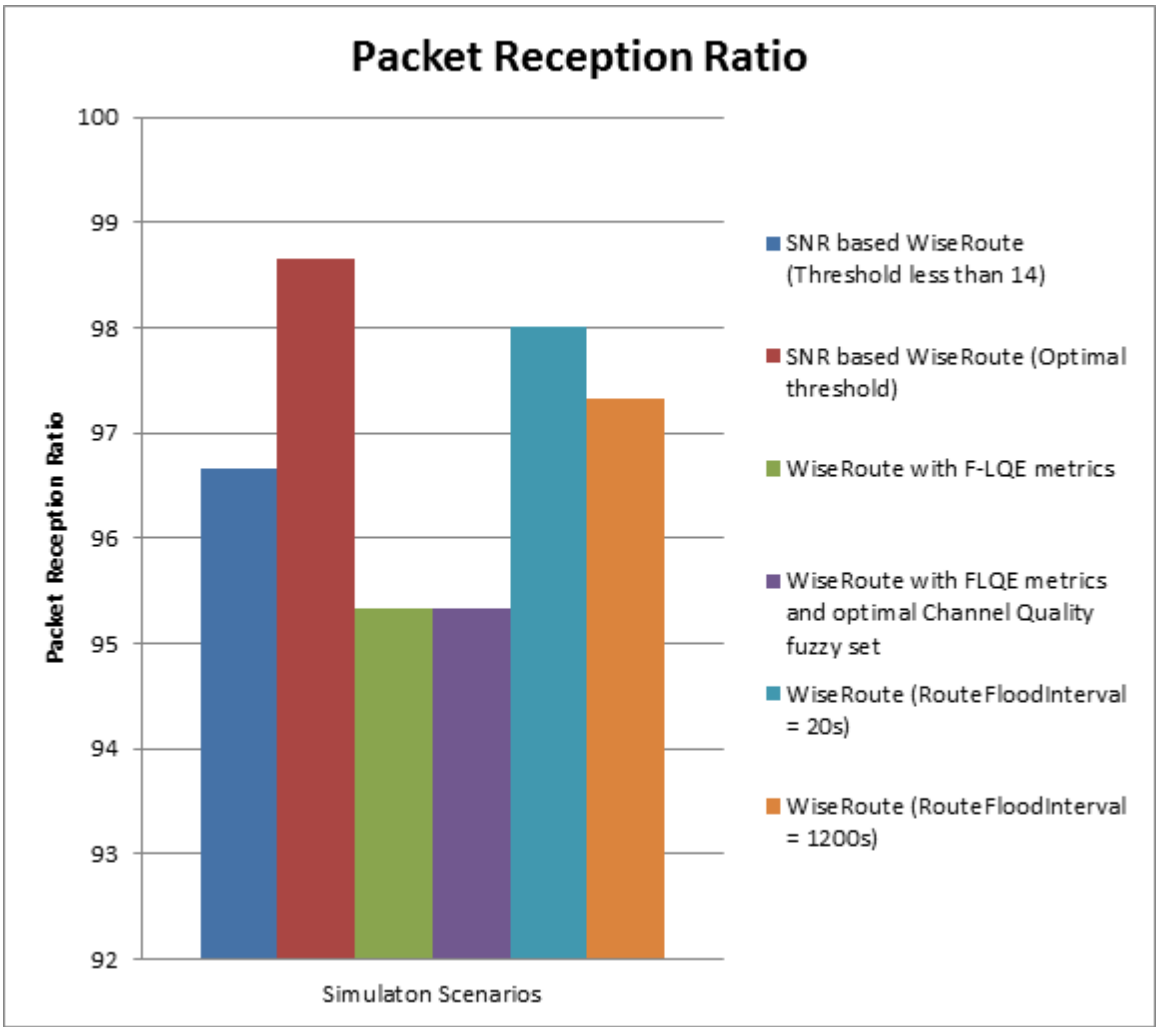

Figure 5.6: Packet Reception Ratio

The second set of experiments consider the mean number of hops taken by a packet to reach sink node and it can be seen that, the scenario 3 and scenario 4, took more hops to transmit data packet than other scenarios. In both scenario 3 and scenario 4, mean number of hops were 10. Other than that scenario 1, scenario 2 (F-LQE-Opt), scenario 5 and scenario 6, all had 8 as their mean number of hops which means that these scenarios were considering the shortest path for data transmission. Figure 5.7 shows mean number of

hops to deliver data packet from source node to sink node in all the simulation scenarios. These experiments like previous experiments show that F-LQE based WiseRoute did not perform as good as any other scenario and the packet which chose F-LQE-Opt was using minimum number of hops to transmit data from source to sink node.

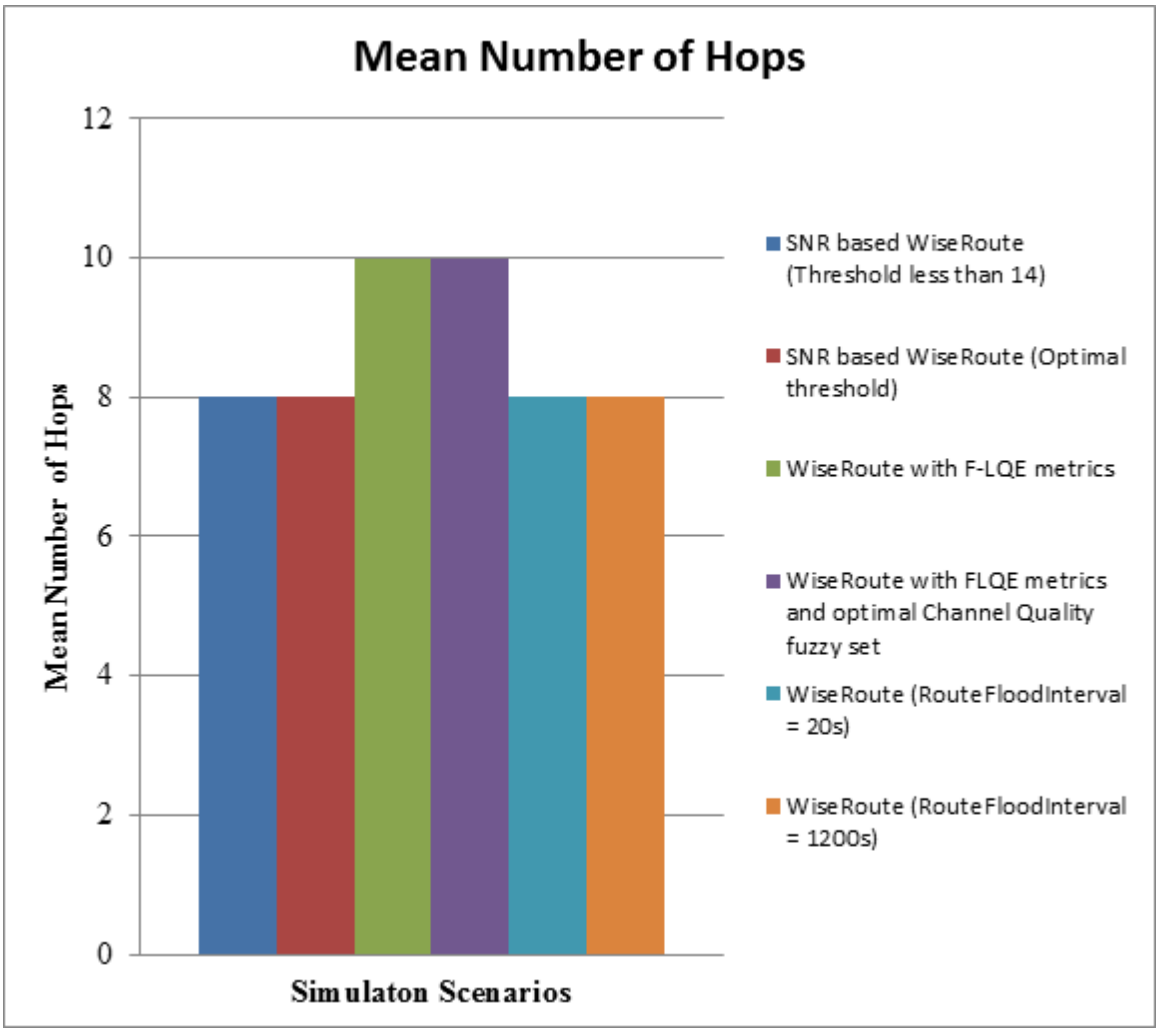

Figure 5.7: Mean Number of Hops

Third set of experiments were considering latency. Figure 5.8 shows mean latency of different network scenarios. Latency was also an important performance metric to

consider as the more time a packet takes to reach sink node, the less efficient the network is. Like previous experiments, these experiments also show that scenario 3 and scenario 4 did not perform as good as any other scenario. It can be seen that scenario 1, scenario 2 (F-LQE-OPT), scenario 5 and scenario 6 had less latency than F-LQE based scenario 3 and scenario 4.

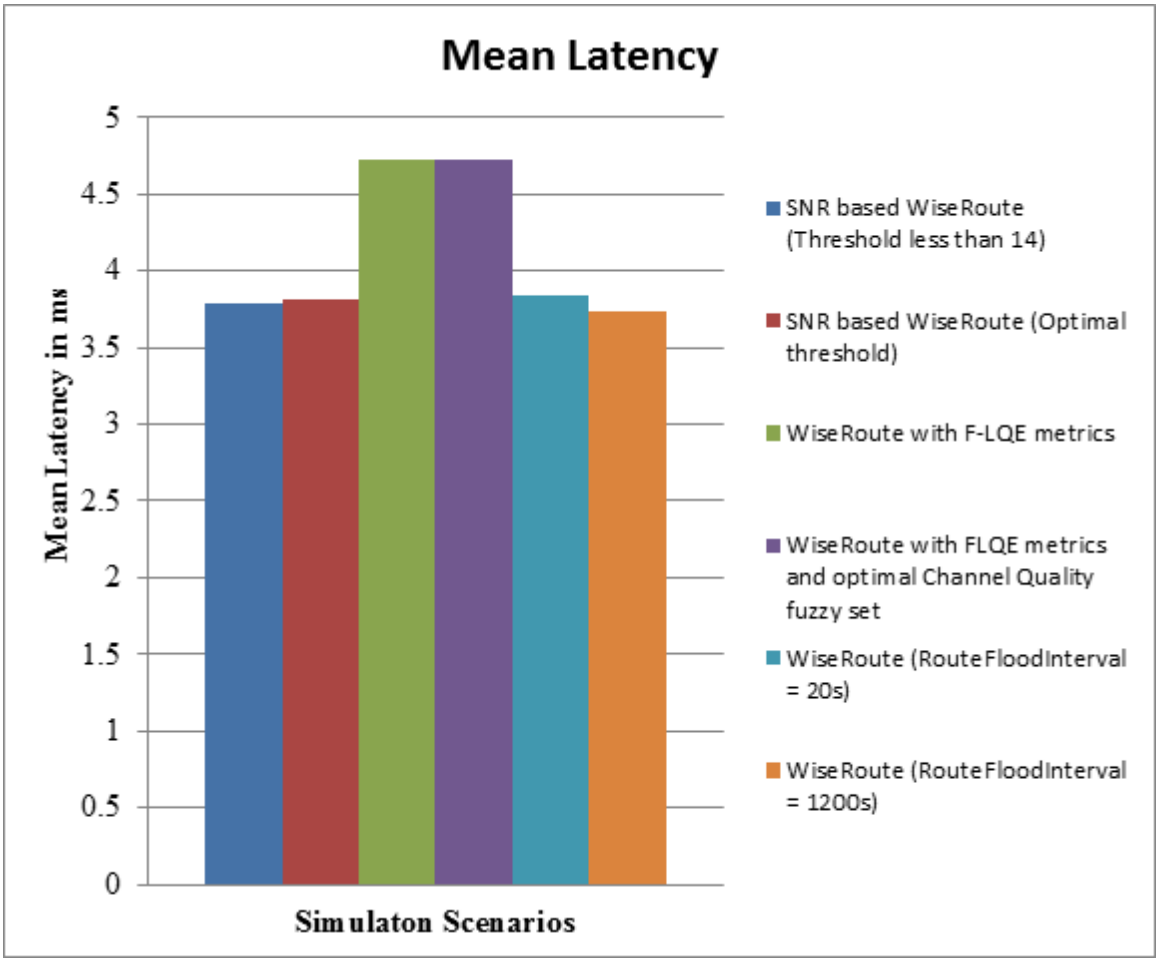

Figure 5.8: Mean Latency

Chapter 6 Conclusion and Future work

The objective of this Master's thesis was to determine the fuzzy sets and several methods were used to calculate the optimum fuzzy membership function for a WSN. A strategy was used to determine the membership function. The fuzzy link estimator was applied in WiseRoute and after a number of simulations a good membership function was assigned and compared to the membership functions used in F-LQE.

In order to evaluate the strategy for selection of refined membership function, fuzzy link estimator was implemented in WiseRoute. It was found that the evaluation in F-LQE was not very accurate as the F-LQE-Opt, which performed better than F-LQE and even RSSI based original WiseRoute. It was found that number of experiments and good knowledge of the domain is needed to evaluate membership function to perform well. The proposed F-LQE-Opt with the new threshold was better in terms of PRR than all scenarios. It was also better than F-LQE based WiseRoute. It also performed better than F-LQE in terms of the mean number of hops taken by each packet to reach sink node.

It should be kept in mind that WiseRoute is a light protocol and the actual implementation in MiXiM is also using a hardware based link estimator so F-LQE-Opt which is also a hardware based metric is performing better.

As future plan fuzzy link estimation in MiXiM can be tested in mobility scenarios and compare the performance to other protocols for mobility and other link estimators. Fuzzy membership functions for other PRR based link quality metrics can also be estimated.

References

[1] A. Cerpa, N. Busek and D. Estrin, SCALE: A Tool for Simple Connectivity Assessment in Lossy Environments, In CENS Technical Report 0021, September 2003.

[2] Kazem Sohraby, Daniel Minoli, and Taieb Znati. Wireless Sensor Networks: Technology, Protocols, and Applications. Wiley-Interscience, 2007.

[3] A. Nayak and I. Stojmenovic, Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordiantion and Data Communications. Hoboken, NJ, USA: John Wiley & Sons, 2010.

[4] Xia, F. QoS Challenges and Opportunities in Wireless Sensor/Actuator Networks. Sensors 2008, 8, 1099-1110.

[5] G. Ács and L. Buttyán, "A Taxonomy of Routing Protocols for Wireless Sensor Networks," Budapest University of Technology and Economics, Hungary, 2007.

[6] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks", Proc. ACM MobiCom 2000, pp.56 -67 2000

[7] Shah, R.C.; Rabaey, J.M., "Energy aware routing for low energy ad hoc sensor networks," Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE , vol.1, no., pp.350,355 vol.1, 17-21 Mar 2002 doi: 10.1109/WCNC.2002.993520

[8] Joanna Kulik, Wendi Heinzelman, and Hari Balakrishnan. 2002. Negotiation-based protocols for disseminating information in wireless sensor networks. Wirel. Netw. 8, 2/3 (March 2002), 169-185.

[9] Arati Manjeshwar; Agrawal, D.P., "TEEN: a routing protocol for enhanced efficiency in wireless sensor networks," Parallel and Distributed Processing Symposium., Proceedings 15th International , vol., no., pp.2009,2015, 23-27 April 2000

[10] Heinzelman, W.R.; Chandrakasan, A.; Balakrishnan, H., "Energy-efficient communication protocol for wireless microsensor networks," System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on , vol., no., pp.10 pp. vol.2,, 4-7 Jan. 2000

[11] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. 2009. Collection tree protocol. In Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09). ACM, New York, NY, USA, 1-14.

[12] Baccour, N.; Koubaa, A.; Ben Jamaa, M.; Youssef, H.; Zuniga, M.; Alves, M., "A comparative simulation study of link quality estimators in wireless sensor networks," Modeling, Analysis & Simulation of Computer and Telecommunication Systems, 2009. MASCOTS '09. IEEE International Symposium on , vol., no., pp.1,10, 21-23 Sept. 2009

[13] Sharma, D.; Liscano, R.; Heydari, S.S., "Collector Tree Protocol (CTP) performance in mobile Wireless Sensor Networks," Electrical and Computer

Engineering (CCECE), 2011 24th Canadian Conference on , vol., no., pp.001548,001552, 8-11 May 2011

[14] R. Fonseca , O. Gnawali , K. Jamieson and P. Levis "Four bit wireless link estimation", Proc. 6th Workshop Hot Topics Netw., 2007

[15] "F-LQE: A fuzzy link quality estimator for wireless sensor networks," in 7th European Conference on Wireless Sensor Networks (EWSN 2010), ser. LNCS 5970. Coimbra, Portugal: Springer, February 2010, p. 240-255.

[16] OMNeT++ Discrete Event Simulation System. Available: http://www.omnetpp.org.

[17] MiXiM simulator for wireless and mobile networks using OMNeT++. [online]. Available: http://mixim.sourceforge.net/.

[18] D. Piguet, J. Rousselot, P. Dallemagne, C. Kassapoglou-Faist "Routing for Mobile Wireless Sensor Networks," CSEM Scientific and Technical Report, 2009, p-26.

[19] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo. TEP 123: The Collection Tree Protocol, Aug. 2006.

[20] S. N. Sivanandam, S. Sumathi, and S. N. Deepa. 2006. Introduction to Fuzzy Logic Using MATLAB. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[21] Boano, C.A.; Zúñiga, M.A.; Voigt, T.; Willig, A.; Romer, K., "The Triangle Metric: Fast Link Quality Estimation for Mobile Wireless Sensor Networks,"

Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on , vol., no., pp.1,7, 2-5 Aug. 2010

[22] Zhi-Qiang Guo; Qin Wang; Mo-Han Li; Jie He, "Fuzzy Logic Based Multidimensional Link Quality Estimation for Multi-Hop Wireless Sensor Networks," Sensors Journal, IEEE , vol.13, no.10, pp.3605,3615, Oct. 2013

[23] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. 2003. A high-throughput path metric for multi-hop wireless routing. In Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom '03). ACM, New York, NY, USA, 134-146.

[24] A. Woo and D. Culler, "Evaluation of efficient link reliability estimators for low-power wireless networks," Tech. Rep., 2003.

[25] T. Winter, P. Thubert, A. Brandt, et al., RPL: IPv6 Routing Protocol for Low power and Lossy Networks, draft-ietf-roll-rpl-19, Internet Engineering Task Force, 2011, http://tools.ietf.org/html/draft-ietf-roll-rpl-19.

[26] Heinzelman, W.R.; Chandrakasan, A.; Balakrishnan, H., "Energy-efficient communication protocol for wireless microsensor networks," System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on , vol., no., pp.10 pp. vol.2,, 4-7 Jan. 2000

[27] JeongGil Ko, Joakim Eriksson, Nicolas Tsiftes, Stephen Dawson-Haggerty, Jean-Philippe Vasseur, Mathilde Durvy, Andreas Terzis, Adam Dunkels, and David Culler. 2011. Industry: beyond interoperability: pushing the performance of

sensor network IP stacks. In Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys '11). ACM, New York, NY, USA, 1-11

[28] F. Karray and C. De Silva Soft Computing and Intelligent Systems Design, 2004 :Addison-Wesley

[29] L. A. Zadeh "Fuzzy sets", Inform. Contr., vol. 8, pp.338 -353 1965

Appendix A

A.1 Algorithm for Fuzzification of Smoothened Packet Reception Ratio (SPRR)

Algorithm 1: Fuzzification of Smoothened Packet Reception Ratio (SPRR)

SET x to PRR value

SET $\boldsymbol{x_1}$TO x-axis lower bound of membership function

SET $\boldsymbol{x_2}$TO x-axis upper bound of membership function

SET $\boldsymbol{y_1}$TO 0 as initial point

SET $\boldsymbol{y_2}$TO 1 as final point

COMPUTE Fuzzy value of TO $\boldsymbol{y} = \frac{(y_2 - y_1) \times (x - x_1)}{y_1 \times (x_2 - x_1)}$

if (x<=$\boldsymbol{x_1}$)

SEND fuzzyfied value of PRR as y=0;

else if (x>=$\boldsymbol{x_2}$)

SEND fuzzyfied value of PRR as y=1

else

SEND fuzzyfied value of PRR as y

A.2 Algorithm for Fuzzification of Asymmetry Level(ASL)

Algorithm 2: Fuzzification of Asymmetry Level(ASL)

SET x to ASL value

SET $\boldsymbol{x_1}$TO x-axis lower bound of membership function

SET $\boldsymbol{x_2}$TO x-axis upper bound of membership function

SET $\boldsymbol{y_1}$TO 1 as initial point

SET $\boldsymbol{y_2}$TO 0 as final point

COMPUTE Fuzzy value of TO $\boldsymbol{y} = \frac{(y_2 - y_1) \times (x - x_1)}{y_1 \times (x_2 - x_1)}$

if (x<=$\boldsymbol{x_1}$)

SEND fuzzyfied value of ASL as y=0;

else if (x>=$\boldsymbol{x_2}$)

SEND fuzzyfied value of ASL as y=1

else

SEND fuzzyfied value of ASL as y

A.3 Algorithm for Fuzzification of Signal to Noise Ratio (SNR)

Algorithm 3: Fuzzification of Signal to Noise Ratio (SNR)

SET x to SNR value

SET $\boldsymbol{x_1}$TO x-axis lower bound of membership function

SET $\boldsymbol{x_2}$TO x-axis upper bound of membership function

SET $\boldsymbol{y_1}$TO 0 as initial point

SET $\boldsymbol{y_2}$TO 1 as final point

COMPUTE Fuzzy value of TO $\boldsymbol{y = \frac{(y_2 - y_1) \times (x - x_1)}{y_1 \times (x_2 - x_1)}}$

if (x<=$\boldsymbol{x_1}$)

SEND fuzzyfied value of SNR as y=0;

else if (x>=$\boldsymbol{x_2}$)

SEND fuzzyfied value of SNR as y=1

else

SEND fuzzyfied value of SNR as y

A.4 Algorithm for Fuzzification of Signal to Noise Ratio (SNR)

Algorithm 4: Fuzzification of Stability Factor(SF)

SET x to SF value

SET $\boldsymbol{x_1}$TO x-axis lower bound of membership function

SET $\boldsymbol{x_2}$TO x-axis upper bound of membership function

SET $\boldsymbol{y_1}$TO 1 as initial point

SET $\boldsymbol{y_2}$TO 0 as final point

COMPUTE Fuzzy value of TO $\boldsymbol{y = \frac{(y_2 - y_1) \times (x - x_1)}{y_1 \times (x_2 - x_1)}}$

if (x<=$\boldsymbol{x_1}$)

SEND fuzzyfied value of SF as y=0;

else if (x>=$\boldsymbol{x_2}$)

SEND fuzzyfied value of SF as y=1

else

SEND fuzzyfied value of SF as y

Appendix B

```
#include "WiseRoute.h"

#include <limits>
#include <algorithm>
#include <cassert>

#include "NetwControlInfo.h"
#include "MacToNetwControlInfo.h"
#include "ArpInterface.h"
#include "FindModule.h"
#include "WiseRoutePkt_m.h"
#include "SimTracer.h"
#include "math.h"
#include "connectionManager/ConnectionManagerAccess.h"

using std::make_pair;

Define_Module(WiseRoute);

void WiseRoute::initialize(int stage)
{
    BaseNetwLayer::initialize(stage);

    if(stage == 1) {

        EV << "Host index=" << findHost()->getIndex() << ", Id="
        << findHost()->getId() << endl;


        EV << "  host IP address=" << myNetwAddr << endl;
        EV << "  host macaddress=" << arp->getMacAddr(myNetwAddr) << endl;
        macaddress = arp->getMacAddr(myNetwAddr);

        sinkAddress = LAddress::L3Type( par("sinkAddress").longValue() ); // 0
        sourceAddress = LAddress::L3Type( par("sourceAddress").longValue() );
// 0

        headerLength = par ("headerLength");
        rssiThreshold = par("rssiThreshold").doubleValue();
        rssiThreshold = FWMath::dBm2mW(rssiThreshold);
        routeFloodsInterval = par("routeFloodsInterval");

        stats = par("stats");
        trace = par("trace");
        debug = par("debug");
        useSimTracer = par("useSimTracer");
        floodSeqNumber = 0;

        nbDataPacketsForwarded = 0;
        nbDataPacketsReceived = 0;
        nbDataPacketsSent = 0;
```

```
        nbDuplicatedFloodsReceived = 0;
        nbFloodsSent = 0;
        nbPureUnicastSent = 0;
        nbRouteFloodsSent = 0;
        nbRouteFloodsReceived = 0;
        nbUnicastFloodForwarded = 0;
        nbPureUnicastForwarded = 0;
        nbGetRouteFailures = 0;
        nbRoutesRecorded = 0;
        nbHops = 0;
        toHops=0;
        holder=0;
        holder1=0;

        counter=0;
        rrflood=0;
        prr=0;
        mean=0;
        bta=0.6;
        var=0;
        std=0;
        sum=0;
        sprr=0;
        sf=0;
        a=0;
        fSprr=0;
        fSf=0;
        floodNumber=0;
        fSnr=0;
        flqe=0;
        lq=0;
        replysent=0;
        replyreceived=0;
        replycounter=0;
        replyprr=0;
        globalflqe=0;
        seq=0;
        cntr=0;
        for(int i=0;i<1000;i++)
        {
            prrarray[i]=0;
            lastlq[i]=0;
            arr[i]=0;
        }
        receivedRSSI.setName("receivedRSSI");
        routeRSSI.setName("routeRSSI");
        allReceivedRSSI.setName("allReceivedRSSI");
        receivedBER.setName("receivedBER");
        routeBER.setName("routeBER");
        allReceivedSNR.setName("allReceivedSNR");
        allReceivedBER.setName("allReceivedBER");
        nextHopSelectionForSink.setName("nextHopSelectionForSink");

        routeFloodTimer = new cMessage("route-flood-timer",
SEND_ROUTE_FLOOD_TIMER);
```

```
        // only schedule a flood of the node is a sink!!
        if (routeFloodsInterval > 0 && myNetwAddr==sinkAddress)
            scheduleAt(simTime() + uniform(0.5, 1.5), routeFloodTimer);

        if(useSimTracer) {
          // Get a handle to the tracer module
          tracer = FindModule<SimTracer*>::findGlobalModule();
          //const char *tracerModulePath = "sim.simTracer";
          //cModule *modp = simulation.getModuleByPath(tracerModulePath);
          //tracer = check_and_cast<SimTracer *>(modp);
          if (!tracer) {
            error("No SimTracer module found, please check your ned
configuration.");
          }
          // log node position
          ChannelMobilityPtrType ptrMobility =
ChannelMobilityAccessType().get();
          if (ptrMobility) {
            Coord pos = ptrMobility->getCurrentPosition();
            tracer->logPosition(myNetwAddr, pos.x, pos.y, pos.z);
          }
        }
    }
}

WiseRoute::~WiseRoute()
{
    cancelAndDelete(routeFloodTimer);
}

void WiseRoute::handleSelfMsg(cMessage* msg)
{
    if (msg->getKind() == SEND_ROUTE_FLOOD_TIMER) {
        // Send route flood packet and restart the timer
        WiseRoutePkt* pkt = new WiseRoutePkt("route-flood", ROUTE_FLOOD);
        pkt->setByteLength(headerLength);
        pkt->setInitialSrcAddr(myNetwAddr);
        pkt->setFinalDestAddr(LAddress::L3BROADCAST);
        pkt->setSrcAddr(myNetwAddr);
        pkt->setDestAddr(LAddress::L3BROADCAST);
        pkt->setNbHops(0);
        pkt->setminFlqe(0);
        floodTable.insert(make_pair(myNetwAddr, floodSeqNumber));
        pkt->setSeqNum(floodSeqNumber);
        floodSeqNumber++;
        pkt->setIsFlood(1);
        pkt->setglobalFlqe(0);
        setDownControlInfo(pkt, LAddress::L2BROADCAST);
        sendDown(pkt);
        nbFloodsSent++;
        nbRouteFloodsSent++;
        if(simTime()<240)
        {
            scheduleAt(simTime() + routeFloodsInterval, routeFloodTimer);
        }
```

```
        //else
         //   scheduleAt(simTime() + 100, routeFloodTimer);
    }
    else {
        EV << "WiseRoute - handleSelfMessage: got unexpected message of kind "
<< msg->getKind() << endl;
        delete msg;
    }
}


void WiseRoute::handleLowerMsg(cMessage* msg)  // This part receives message
from other nodes like data packets
{
    WiseRoutePkt*           netwMsg        =
check_and_cast<WiseRoutePkt*>(msg);
    const LAddress::L3Type& finalDestAddr  = netwMsg->getFinalDestAddr();
    const LAddress::L3Type& initialSrcAddr = netwMsg->getInitialSrcAddr();
    const LAddress::L3Type& srcAddr        = netwMsg->getSrcAddr();
    double rssi = static_cast<MacToNetwControlInfo*>(netwMsg-
>getControlInfo())->getRSSI();
    double ber = static_cast<MacToNetwControlInfo*>(netwMsg-
>getControlInfo())->getBitErrorRate();
    double snr= static_cast<MacToNetwControlInfo*>(netwMsg->getControlInfo())-
>getSNR();
    // Check whether the message is a flood and if it has to be forwarded.
    floodTypes floodType = updateFloodTable(netwMsg->getIsFlood(),
initialSrcAddr, finalDestAddr,
                                            netwMsg->getSeqNum());

    if(netwMsg->getKind()==DATA)
    {
        EV<<"\n\n\n\n\nNode "<<myNetwAddr<<" Received Packet from "<<netwMsg-
>getSrcAddr();
    }

    if(netwMsg->getKind()==REPLY)
    {
        if(netwMsg->getDestAddr()==myNetwAddr)
        {
            replycounter=netwMsg->getSeqNum()+1;
            replyreceived++;
        }
        delete netwMsg;
    }
    else {
        const cObject* pCtrlInfo = NULL;
        // If the message is a route flood, update the routing table.
if (netwMsg->getKind() == ROUTE_FLOOD)
        {
            fSnr=funcSNR(snr);

            flqe=0;
            lq=0;
            flqe=(bta*fSnr)+((1-bta)*fSnr);        // check this code
```

```
            lq=100*flqe;
            if(netwMsg->getminFlqe()<=0)
            {
                netwMsg->setminFlqe(lq);
            }
            if(lq<netwMsg->getminFlqe())
            {
                netwMsg->setminFlqe(lq);
            }
            lq+=netwMsg->getglobalFlqe();
            EV<<"\n\n\n\n\nFor SNR "<<snr<<"Fuzzified FSNR is "<<fSnr<<" LQ is
"<<lq<<" FLQE is "<<flqe;
            updateRouteTable(netwMsg, lq);

            WiseRoutePkt*    reply   = new WiseRoutePkt("REPLY", REPLY);
            reply->setByteLength(headerLength);
            reply->setInitialSrcAddr(myNetwAddr);
            reply->setFinalDestAddr(netwMsg->getSrcAddr());
            reply->setSrcAddr(myNetwAddr);
            reply->setIsFlood(0);
            floodNumber++;
            reply->setSeqNum(floodNumber);
            reply->setDestAddr(netwMsg->getSrcAddr());
            replysent++;
            setDownControlInfo(reply, (arp->getMacAddr(netwMsg-
>getSrcAddr())));
            sendDown(reply);
        }


        if (finalDestAddr == myNetwAddr ||
LAddress::isL3Broadcast(finalDestAddr)) {
            WiseRoutePkt* msgCopy;
            if (floodType == FORWARD) { // data packet won't come here
                msgCopy = check_and_cast<WiseRoutePkt*>(netwMsg->dup());
                netwMsg->setSrcAddr(myNetwAddr);
                floodNumber++;
                pCtrlInfo = netwMsg->removeControlInfo();
                setDownControlInfo(netwMsg, LAddress::L2BROADCAST);
                netwMsg->setNbHops(netwMsg->getNbHops()+1);
                netwMsg->setglobalFlqe(netwMsg->getglobalFlqe()+lq);
                sendDown(netwMsg);
                arr[cntr]=netwMsg->getSeqNum();
                nbDataPacketsForwarded++;
                EV<<"\n\n\n\n\n\n\nSNR for Node "<<myNetwAddr<<" is "<<snr;
            }
            else
                msgCopy = netwMsg;
            if (msgCopy->getKind() == DATA) { //This part is used when data
packet is 1 hop away from node 0 or destination
                sendUp(decapsMsg(msgCopy));
                nbDataPacketsReceived++;
            }
            else {
```

```
                nbRouteFloodsReceived++;// this part is used when data packet
is forworded
                delete msgCopy;
            }
        }
        else {
            // not for me. if flood, forward as flood. else select a route
            if (floodType == FORWARD) {
                netwMsg->setSrcAddr(myNetwAddr);
                pCtrlInfo = netwMsg->removeControlInfo();
                setDownControlInfo(netwMsg, LAddress::L2BROADCAST);
                netwMsg->setNbHops(netwMsg->getNbHops()+1);
                sendDown(netwMsg);
                nbDataPacketsForwarded++;
                nbUnicastFloodForwarded++;
            }
            else { // this part is used when data packet is forwarded to
different hops towards 0 or sink
                LAddress::L3Type nextHop = getRoute(finalDestAddr);
                if (LAddress::isL3Broadcast(nextHop)) {
                    // no route exist to destination, attempt to send to final
destination
                    nextHop = finalDestAddr;
                    nbGetRouteFailures++;
                }

                netwMsg->setSrcAddr(myNetwAddr);
                netwMsg->setDestAddr(nextHop);
                pCtrlInfo = netwMsg->removeControlInfo();
                setDownControlInfo(netwMsg, arp->getMacAddr(nextHop));
                netwMsg->setNbHops(netwMsg->getNbHops()+1);
                sendDown(netwMsg);
                nbDataPacketsForwarded++;
                nbPureUnicastForwarded++;
            }
        }
        if (pCtrlInfo != NULL)
            delete pCtrlInfo;
    }
}

void WiseRoute::handleLowerControl(cMessage *msg)
{
    delete msg;
}

void WiseRoute::handleUpperMsg(cMessage* msg)
{
    LAddress::L3Type finalDestAddr;
    LAddress::L3Type nextHopAddr;
    LAddress::L2Type nextHopMacAddr;
    WiseRoutePkt*    pkt   = new WiseRoutePkt(msg->getName(), DATA);
    cObject*         cInfo = msg->removeControlInfo();

    pkt->setByteLength(headerLength);
```

```
    if ( cInfo == NULL ) {
        EV << "WiseRoute warning: Application layer did not specifiy a
destination L3 address\n"
           << "\tusing broadcast address instead\n";
        finalDestAddr = LAddress::L3BROADCAST;  //weird
    }
    else {
        EV <<"WiseRoute: CInfo removed, netw addr="<<
NetwControlInfo::getAddressFromControlInfo( cInfo ) <<endl;
        finalDestAddr = NetwControlInfo::getAddressFromControlInfo( cInfo );
        delete cInfo;
    }

    pkt->setFinalDestAddr(finalDestAddr);
    pkt->setInitialSrcAddr(myNetwAddr);// weired
    pkt->setSrcAddr(myNetwAddr);
    pkt->setNbHops(0); //weird

    if (LAddress::isL3Broadcast(finalDestAddr))
        nextHopAddr = LAddress::L3BROADCAST;
    else
        nextHopAddr = getRoute(finalDestAddr, true);
    pkt->setDestAddr(nextHopAddr);
    if (LAddress::isL3Broadcast(nextHopAddr)) {
        // it's a flood.
        nextHopMacAddr = LAddress::L2BROADCAST;
        pkt->setIsFlood(1);
        nbFloodsSent++;
        // record flood in flood table
        floodTable.insert(make_pair(myNetwAddr, floodSeqNumber));
        pkt->setSeqNum(floodSeqNumber);
        floodSeqNumber++;
        nbGetRouteFailures++;
    }
    else {
        pkt->setIsFlood(0);
        nbPureUnicastSent++;
        nextHopMacAddr = arp->getMacAddr(nextHopAddr);
    }
    setDownControlInfo(pkt, nextHopMacAddr);
    assert(static_cast<cPacket*>(msg));
    pkt->encapsulate(static_cast<cPacket*>(msg));
    sendDown(pkt);
    nbDataPacketsSent++;
}

void WiseRoute::finish()
{
    if (stats) {
        recordScalar("nbDataPacketsForwarded", nbDataPacketsForwarded);
        recordScalar("nbDataPacketsReceived", nbDataPacketsReceived);
        recordScalar("nbDataPacketsSent", nbDataPacketsSent);
        recordScalar("nbDuplicatedFloodsReceived",
nbDuplicatedFloodsReceived);
```

```
        recordScalar("nbFloodsSent", nbFloodsSent);
        recordScalar("nbPureUnicastSent", nbPureUnicastSent);
        recordScalar("nbRouteFloodsSent", nbRouteFloodsSent);
        recordScalar("nbRouteFloodsReceived", nbRouteFloodsReceived);
        recordScalar("nbUnicastFloodForwarded", nbUnicastFloodForwarded);
        recordScalar("nbPureUnicastForwarded", nbPureUnicastForwarded);
        recordScalar("nbGetRouteFailures", nbGetRouteFailures);
        recordScalar("nbRoutesRecorded", nbRoutesRecorded);
        recordScalar("meanNbHops", (double) nbHops / (double)
nbDataPacketsReceived);
    }
}

void WiseRoute::updateRouteTable(WiseRoutePkt *netwMsg, double flqi){
    tRouteTable::iterator pos;

    toHops=netwMsg->getminFlqe();
    lastlq[seq]=(0.7*(flqi/(netwMsg->getNbHops()+1)))+((1-0.7)*toHops);
    EV<<"\n\n\n\n\n\n\nAT node "<<myNetwAddr<<"the final flqe is
"<<lastlq[seq];
    seq++;
    pos = routeTable.find(netwMsg->getInitialSrcAddr());


    if(seq>1 && myNetwAddr==29)
        {
            tRouteTableEntry newEntry;
            // last hop from origin means next hop towards origin.
            if (lastlq[seq-1]>lastlq[seq-2])// && lastlq[seq-1]>lastlq[seq-3])
            {
            newEntry.nextHop = netwMsg->getSrcAddr();
            routeTable.insert(make_pair(netwMsg->getInitialSrcAddr(),
newEntry));
            nbRoutesRecorded++;
            if (netwMsg->getInitialSrcAddr() == LAddress::L3NULL && trace)
            {
                nextHopSelectionForSink.record(static_cast<double>(netwMsg-
>getSrcAddr()));
            }
            }
    }
    else if (myNetwAddr!=29)// && pos == routeTable.end())
    {
        if (lastlq[seq-1]>lastlq[seq-2] && netwMsg->getSrcAddr()!=29)// &&
lastlq[seq-1]>lastlq[seq-3])
        {
        tRouteTableEntry newEntry;
        newEntry.nextHop = netwMsg->getSrcAddr();
        routeTable.insert(make_pair(netwMsg->getInitialSrcAddr(), newEntry));
        nbRoutesRecorded++;
        if (netwMsg->getInitialSrcAddr() == LAddress::L3NULL && trace)
        {
            nextHopSelectionForSink.record(static_cast<double>(netwMsg-
>getSrcAddr()));
        }
```

```
        }
    }

}

cMessage* WiseRoute::decapsMsg(WiseRoutePkt *msg)
{
    cMessage *m = msg->decapsulate();
    setUpControlInfo(m, msg->getSrcAddr());
    nbHops = nbHops + msg->getNbHops();
    // delete the netw packet
    delete msg;
    return m;
}

WiseRoute::floodTypes WiseRoute::updateFloodTable(bool isFlood, const
tFloodTable::key_type& srcAddr, const tFloodTable::key_type& destAddr,
unsigned long seqNum)
{
    if (isFlood) {
        tFloodTable::iterator pos = floodTable.lower_bound(srcAddr);
        tFloodTable::iterator posEnd = floodTable.upper_bound(srcAddr);

        while (pos != posEnd && holder==srcAddr) {
            if (seqNum == pos->second)
                    return DUPLICATE;  // this flood is known, don't forward
it.
            ++pos;
            holder=srcAddr;
        }
        floodTable.insert(make_pair(srcAddr, seqNum));
        if (destAddr == myNetwAddr)
            return FORME;
        else
            return FORWARD;
    }
    else
        return NOTAFLOOD;
    holder=srcAddr;
}

WiseRoute::tFloodTable::key_type WiseRoute::getRoute(const
tFloodTable::key_type& destAddr, bool /*iAmOrigin*/) const
{
    tRouteTable::const_iterator pos = routeTable.find(destAddr);
    if (pos != routeTable.end())
        return pos->second.nextHop;
   // else
   //     return LAddress::L3BROADCAST;
}



double WiseRoute::funcSPRR(double var1)
```

```
{
    double fval=0;
    double x=0;
    double y=0;
    x=var1;
    y=(x-0.45)/0.5; // y1=0 so I removed the values of y1 from y=((y2-y1)*(x-
x1))/(y1*(x2-x1))
    if (x<=0.45)
        fval=0;// y=0;
    else if (x>=0.95)
        fval=1;// y=1;
    else if(y>1)
        fval= 1;
    else if(y<0)
        fval= 0;
    else
        fval= y;
    return fval;
}

double WiseRoute::funcSF(double var2)
{
    double fval=0;
    double x=0;
    double y=0;
    x=var2;
    y=((-1*x)/0.7)+1;
        fval= 1;
    else if (x>=0.7)
        fval= 0;
    else if(y>1)
        fval= 1;
    else if(y<0)
        fval= 0;
    else
        fval= y;
    return fval;
}


double WiseRoute::funcSNR(double var3)
{
    double fval=0;
    double x=0;
    double y=0;
    x=var3;
    y= (x-1)/7;
    if (x<=1)
        fval= 0;
    else if (x>=8)
        fval= 1;
    else if(y>1)
        fval= 1;
    else if(y<0)
        fval= 0;
```

```
    else
        fval= y;
    return fval;
}

double WiseRoute::funcASL(double var4)
{
    double fval=0;
    double x=0;
    double y=0;
    x=var4;
    y=((-1*(x-0.1))/(0.4))+1;
    if (x<=0.1)
        fval= 1;
    else if (x>=0.5)
        fval= 0;
    else if(y>1)
        fval= 1;
    else if(y<0)
        fval= 0;
    else
        fval= y;
    return fval;
}
```